

Oracle® Big Data Appliance

Software User's Guide

Release 4 (4.1)

E57351-03

February 2015

Describes the Oracle Big Data Appliance software available to administrators and software developers.

Copyright © 2011, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Cloudera, Cloudera CDH, and Cloudera Manager are registered and unregistered trademarks of Cloudera, Inc.

Contents

Preface	ix
Audience	ix
Documentation Accessibility	ix
Related Documents	ix
Conventions	x
Backus-Naur Form Syntax	x
 1 Introducing Oracle Big Data Appliance	
What Is Big Data?	1-1
High Variety	1-1
High Complexity	1-2
High Volume	1-2
High Velocity	1-2
The Oracle Big Data Solution	1-2
Software for Big Data	1-3
Software Component Overview	1-4
Acquiring Data for Analysis	1-5
Hadoop Distributed File System	1-5
Apache Hive	1-5
Oracle NoSQL Database	1-5
Organizing Big Data	1-6
MapReduce	1-7
Oracle Big Data SQL	1-7
Oracle Big Data Connectors	1-7
Oracle R Support for Big Data	1-9
Analyzing and Visualizing Big Data	1-9
 2 Administering Oracle Big Data Appliance	
Monitoring Multiple Clusters Using Oracle Enterprise Manager	2-1
Using the Enterprise Manager Web Interface	2-1
Using the Enterprise Manager Command-Line Interface	2-2
Managing Operations Using Cloudera Manager	2-3
Monitoring the Status of Oracle Big Data Appliance	2-3
Performing Administrative Tasks	2-4
Managing CDH Services With Cloudera Manager	2-4

Using Hadoop Monitoring Utilities	2-5
Monitoring MapReduce Jobs.....	2-5
Monitoring the Health of HDFS.....	2-6
Using Cloudera Hue to Interact With Hadoop	2-6
About the Oracle Big Data Appliance Software	2-7
Software Components	2-8
Unconfigured Software	2-9
Allocating Resources Among Services.....	2-10
About the CDH Software Services	2-10
Where Do the Services Run on a Single-Rack CDH Cluster?	2-10
Where Do the Services Run on a Multirack CDH Cluster?	2-11
About MapReduce	2-12
Automatic Failover of the NameNode.....	2-12
Automatic Failover of the ResourceManager	2-13
Map and Reduce Resource Configuration	2-14
Effects of Hardware on Software Availability	2-15
Logical Disk Layout	2-15
Critical and Noncritical CDH Nodes	2-15
First NameNode Node	2-16
Second NameNode Node	2-17
First ResourceManager Node.....	2-17
Second ResourceManager Node	2-17
Noncritical CDH Nodes	2-17
Managing a Hardware Failure	2-18
About Oracle NoSQL Database Clusters.....	2-18
Prerequisites for Managing a Failing Node	2-18
Managing a Failing CDH Critical Node	2-19
Managing a Failing Noncritical Node.....	2-19
Stopping and Starting Oracle Big Data Appliance.....	2-20
Prerequisites.....	2-20
Stopping Oracle Big Data Appliance	2-20
Starting Oracle Big Data Appliance.....	2-22
Managing Oracle Big Data SQL	2-24
Adding and Removing the Oracle Big Data SQL Service	2-24
Allocating Resources to Oracle Big Data SQL	2-24
Security on Oracle Big Data Appliance	2-25
About Predefined Users and Groups	2-26
About User Authentication.....	2-26
About Fine-Grained Authorization	2-27
About On-Disk Encryption.....	2-27
Port Numbers Used on Oracle Big Data Appliance.....	2-27
About Puppet Security	2-28
Auditing Oracle Big Data Appliance	2-28
About Oracle Audit Vault and Database Firewall	2-29
Setting Up the Oracle Big Data Appliance Plug-in	2-29
Monitoring Oracle Big Data Appliance	2-30
Collecting Diagnostic Information for Oracle Customer Support	2-31

3 Supporting User Access to Oracle Big Data Appliance

About Accessing a Kerberos-Secured Cluster	3-1
Providing Remote Client Access to CDH	3-2
Prerequisites.....	3-2
Installing a CDH Client on Any Supported Operating System	3-3
Configuring a CDH Client for an Unsecured Cluster	3-3
Configuring a CDH Client for a Kerberos-Secured Cluster	3-4
Verifying Access to a Cluster from the CDH Client	3-5
Providing Remote Client Access to Hive.....	3-6
Managing User Accounts	3-8
Creating Hadoop Cluster Users.....	3-8
Providing User Login Privileges (Optional)	3-9
Recovering Deleted Files	3-10
Restoring Files from the Trash	3-10
Changing the Trash Interval.....	3-11
Disabling the Trash Facility	3-11

4 Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance

About Optimizing Communications.....	4-1
About Applications that Pull Data Into Oracle Exadata Database Machine.....	4-1
About Applications that Push Data Into Oracle Exadata Database Machine	4-2
Prerequisites for Optimizing Communications.....	4-2
Specifying the InfiniBand Connections to Oracle Big Data Appliance.....	4-2
Specifying the InfiniBand Connections to Oracle Exadata Database Machine.....	4-3
Enabling SDP on Exadata Database Nodes	4-4
Creating an SDP Listener on the InfiniBand Network.....	4-5

5 Optimizing MapReduce Jobs Using Perfect Balance

What is Perfect Balance?	5-1
About Balancing Jobs Across Map and Reduce Tasks	5-2
Ways to Use Perfect Balance Features.....	5-2
Perfect Balance Components	5-2
Application Requirements	5-2
Getting Started with Perfect Balance	5-3
Analyzing a Job's Reducer Load.....	5-4
About Job Analyzer	5-4
Running Job Analyzer as a Standalone Utility	5-4
Running Job Analyzer Using Perfect Balance.....	5-6
Reading the Job Analyzer Report	5-8
About Configuring Perfect Balance.....	5-9
Running a Balanced MapReduce Job Using Perfect Balance	5-10
About Perfect Balance Reports	5-12
About Chopping.....	5-13
Selecting a Chopping Method	5-13
How Chopping Impacts Applications	5-14

Troubleshooting Jobs Running with Perfect Balance	5-14
Using the Perfect Balance API	5-15
Modifying Your Java Code to Use Perfect Balance.....	5-15
Running Your Modified Java Code with Perfect Balance	5-16
About the Perfect Balance Examples	5-17
About the Examples in This Chapter	5-17
Extracting the Example Data Set.....	5-18
Perfect Balance Configuration Property Reference	5-18

6 Using Oracle Big Data SQL for Data Access

What Is Oracle Big Data SQL?	6-1
About Oracle External Tables.....	6-2
About the Access Drivers for Oracle Big Data SQL	6-2
About Smart Scan Technology	6-2
About Data Security with Oracle Big Data SQL	6-2
Installing Oracle Big Data SQL	6-3
Prerequisites for Using Oracle Big Data SQL 1.1.....	6-3
Performing the Installation.....	6-3
Running the Post-Installation Script for Oracle Big Data SQL.....	6-4
Creating an Oracle External Table for Hive Data	6-5
Obtaining Information About a Hive Table.....	6-5
Using the CREATE_EXTDDL_FOR_HIVE Function.....	6-6
Developing a CREATE TABLE Statement for ORACLE_HIVE	6-7
Creating an Oracle External Table for Oracle NoSQL Database	6-8
Creating a Hive External Table for Oracle NoSQL Database	6-8
Creating the Oracle Database Table for Oracle NoSQL Data	6-9
About Column Data Type Mappings.....	6-10
Example of Accessing Data in Oracle NoSQL Database	6-10
Creating an Oracle External Table for Apache HBase	6-13
Creating a Hive External Table for HBase.....	6-13
Creating the Oracle Database Table for HBase.....	6-14
Creating an Oracle External Table for HDFS Files	6-14
Using the Default Access Parameters with ORACLE_HDFS.....	6-14
Overriding the Default ORACLE_HDFS Settings.....	6-15
About the SQL CREATE TABLE Statement	6-16
Basic Syntax.....	6-16
About the External Table Clause	6-16
About Data Type Conversions	6-18
Querying External Tables	6-19
Granting User Access	6-19
About Error Handling.....	6-19
About the Log Files.....	6-20
About Oracle Big Data SQL on Oracle Exadata Database Machine	6-20
Starting and Stopping the Big Data SQL Agent	6-20
About the Common Directory	6-20
Common Configuration Properties.....	6-20
About the Cluster Directory	6-23

About Permissions	6-23
7 Oracle Big Data SQL Reference	
DBMS_HADOOP PL/SQL Package	7-2
CREATE TABLE ACCESS PARAMETERS Clause	7-5
Static Data Dictionary Views for Hive.....	7-23
8 Copying Oracle Tables to Hadoop	
What Is Copy to BDA?.....	8-1
Getting Started Using Copy to BDA	8-1
Installing Copy to BDA.....	8-2
Prerequisites for Copy to BDA.....	8-2
Installing Copy to BDA on Oracle Big Data Appliance.....	8-2
Installing Copy to BDA on Oracle Exadata Database Machine	8-2
Generating the Data Pump Files	8-2
About Data Pump Format Files	8-3
Identifying the Target Directory	8-3
About the CREATE TABLE Syntax	8-3
Copying the Files to HDFS	8-4
Creating a Hive Table.....	8-4
About Hive External Tables.....	8-4
About Column Mappings	8-4
About Data Type Conversions.....	8-5
Example Using the Sample Schemas.....	8-5
About the Sample Data	8-5
Creating the EXPDIR Database Directory	8-6
Creating Data Pump Format Files for Customer Data	8-6
Verifying the Contents of the Data Files.....	8-7
Copying the Files into Hadoop	8-7
Creating a Hive External Table	8-8
Querying the Data in Hive.....	8-8

Glossary

Index

Preface

The *Oracle Big Data Appliance Software User's Guide* describes how to manage and use the installed software.

Audience

This guide is intended for users of Oracle Big Data Appliance including:

- Application developers
- Data analysts
- Data scientists
- Database administrators
- System administrators

The *Oracle Big Data Appliance Software User's Guide* introduces Oracle Big Data Appliance installed software, features, concepts, and terminology. However, you must acquire the necessary information about administering Hadoop clusters and writing MapReduce programs from other sources.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit
<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For more information, see the following documents:

- *Oracle Big Data Appliance Perfect Balance Java API Reference*
- *Oracle Enterprise Manager System Monitoring Plug-in Installation Guide for Oracle Big Data Appliance*
- *Oracle Big Data Appliance Owner's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
# prompt	The pound (#) prompt indicates a command that is run as the Linux root user.

Backus-Naur Form Syntax

The syntax in this reference is presented in a simple variation of Backus-Naur Form (BNF) that uses the following symbols and conventions:

Symbol or Convention	Description
[]	Brackets enclose optional items.
{ }	Braces enclose a choice of items, only one of which is required.
	A vertical bar separates alternatives within brackets or braces.
...	Ellipses indicate that the preceding syntactic element can be repeated.
delimiters	Delimiters other than brackets, braces, and vertical bars must be entered as shown.
boldface	Words appearing in boldface are keywords. They must be typed as shown. (Keywords are case-sensitive in some, but not all, operating systems.) Words that are not in boldface are placeholders for which you must substitute a name or value.

Part I

Administration

This part describes Oracle Big Data Appliance and provides instructions for routine administrative tasks. It contains the following chapters:

- [Chapter 1, "Introducing Oracle Big Data Appliance"](#)
- [Chapter 2, "Administering Oracle Big Data Appliance"](#)
- [Chapter 3, "Supporting User Access to Oracle Big Data Appliance"](#)
- [Chapter 4, "Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance"](#)

Introducing Oracle Big Data Appliance

This chapter presents an overview of Oracle Big Data Appliance and describes the software installed on the system. This chapter contains the following sections:

- [What Is Big Data?](#)
- [The Oracle Big Data Solution](#)
- [Software for Big Data](#)
- [Acquiring Data for Analysis](#)
- [Organizing Big Data](#)
- [Analyzing and Visualizing Big Data](#)

What Is Big Data?

Using transactional data as the source of business intelligence has been commonplace for many years. As digital technology and the World Wide Web spread into every aspect of modern life, other sources of data can make important contributions to business decision making. Many businesses are looking to these new data sources. They are finding opportunities in analyzing vast amounts of data that until recently was discarded.

Big data is characterized by:

- A variety of data sources: [High Variety](#)
- A complexity of data types: [High Complexity](#)
- A high volume of data flow: [High Volume](#)
- A high velocity of data transactions: [High Velocity](#)

These characteristics pinpoint the challenges in deriving value from big data, and the differences between big data and traditional data sources that primarily provide highly structured, transactional data.

High Variety

Big data is derived from a variety of sources, such as:

- Equipment sensors: Medical, manufacturing, transportation, and other machine sensor transmissions
- Machines: Call detail records, web logs, smart meter readings, Global Positioning System (GPS) transmissions, and trading systems records

- Social media: Data streams from social media sites such as Facebook and blogging sites such as Twitter

Analysts can mine this data repeatedly as they devise new ways of extracting meaningful insights. What seems irrelevant today might prove to be highly pertinent to your business tomorrow.

Challenge: Delivering flexible systems to handle this high variety

High Complexity

As the variety of data types increases, the complexity of the system increases. The complexity of data types also increases in big data because of its low structure.

Challenge: Finding solutions that apply across a broad range of data types.

High Volume

Social media can generate terabytes of daily data. Equipment sensors and other machines can generate that much data in less than an hour.

Even traditional data sources for data warehouses, such as customer profiles from customer relationship management (CRM) systems, transactional enterprise resource planning (ERP) data, store transactions, and general ledger data, have increased tenfold in volume over the past decade.

Challenge: Providing scalability and ease in growing the system

High Velocity

Huge numbers of sensors, web logs, and other machine sources generate data continuously and at a much higher speed than traditional sources, such as individuals entering orders into a transactional database.

Challenge: Handling the data at high speed without stressing the structured systems

The Oracle Big Data Solution

Oracle Big Data Appliance is an engineered system comprising both hardware and software components. The hardware is optimized to run the enhanced big data software components.

Oracle Big Data Appliance delivers:

- A complete and optimized solution for big data
- Single-vendor support for both hardware and software
- An easy-to-deploy solution
- Tight integration with Oracle Database and Oracle Exadata Database Machine

Oracle provides a big data platform that captures, organizes, and supports deep analytics on extremely large, complex data streams flowing into your enterprise from many data sources. You can choose the best storage and processing location for your data depending on its structure, workload characteristics, and end-user requirements.

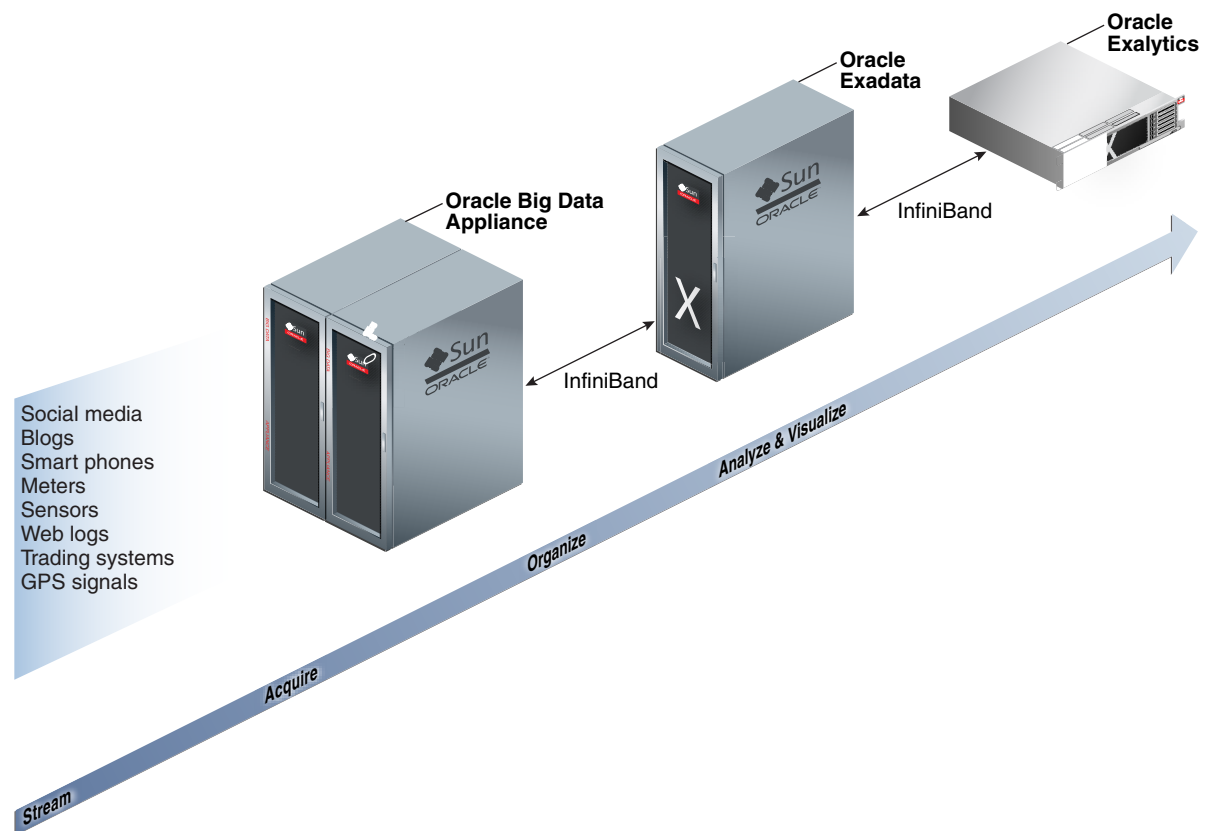
Oracle Database enables all data to be accessed and analyzed by a large user community using identical methods. By adding Oracle Big Data Appliance in front of Oracle Database, you can bring new sources of information to an existing data warehouse. Oracle Big Data Appliance is the platform for acquiring and organizing

big data so that the relevant portions with true business value can be analyzed in Oracle Database.

For maximum speed and efficiency, Oracle Big Data Appliance can be connected to Oracle Exadata Database Machine running Oracle Database. Oracle Exadata Database Machine provides outstanding performance in hosting data warehouses and transaction processing databases. Moreover, Oracle Exadata Database Machine can be connected to Oracle Exalytics In-Memory Machine for the best performance of business intelligence and planning applications. The InfiniBand connections between these engineered systems provide high parallelism, which enables high-speed data transfer for batch or query workloads.

Figure 1–1 shows the relationships among these engineered systems.

Figure 1–1 Oracle Engineered Systems for Big Data



Software for Big Data

The **Oracle Linux** operating system and Cloudera's Distribution including Apache Hadoop (CDH) underlie all other software components installed on Oracle Big Data Appliance. **CDH** is an integrated stack of components that have been tested and packaged to work together.

CDH has a batch processing infrastructure that can store files and distribute work across a set of computers. Data is processed on the same computer where it is stored. In a single Oracle Big Data Appliance rack, CDH distributes the files and workload across 18 servers, which compose a **cluster**. Each server is a node in the cluster.

The software framework consists of these primary components:

- **File system:** The **Hadoop Distributed File System (HDFS)** is a highly scalable file system that stores large files across multiple servers. It achieves reliability by replicating data across multiple servers without RAID technology. It runs on top of the Linux file system on Oracle Big Data Appliance.
- **MapReduce engine:** The **MapReduce** engine provides a platform for the massively parallel execution of algorithms written in Java. Oracle Big Data Appliance 3.0 runs **YARN** by default.
- **Administrative framework:** **Cloudera Manager** is a comprehensive administrative tool for CDH. In addition, you can use Oracle Enterprise Manager to monitor both the hardware and software on Oracle Big Data Appliance.
- **Apache projects:** CDH includes Apache projects for MapReduce and HDFS, such as **Hive**, **Pig**, **Oozie**, **ZooKeeper**, **HBase**, **Sqoop**, and **Spark**.
- **Cloudera applications:** Oracle Big Data Appliance installs all products included in Cloudera Enterprise Data Hub Edition, including **Impala**, **Search**, and **Navigator**.

CDH is written in Java, and Java is the language for applications development. However, several CDH utilities and other software available on Oracle Big Data Appliance provide graphical, web-based, and other language interfaces for ease of use.

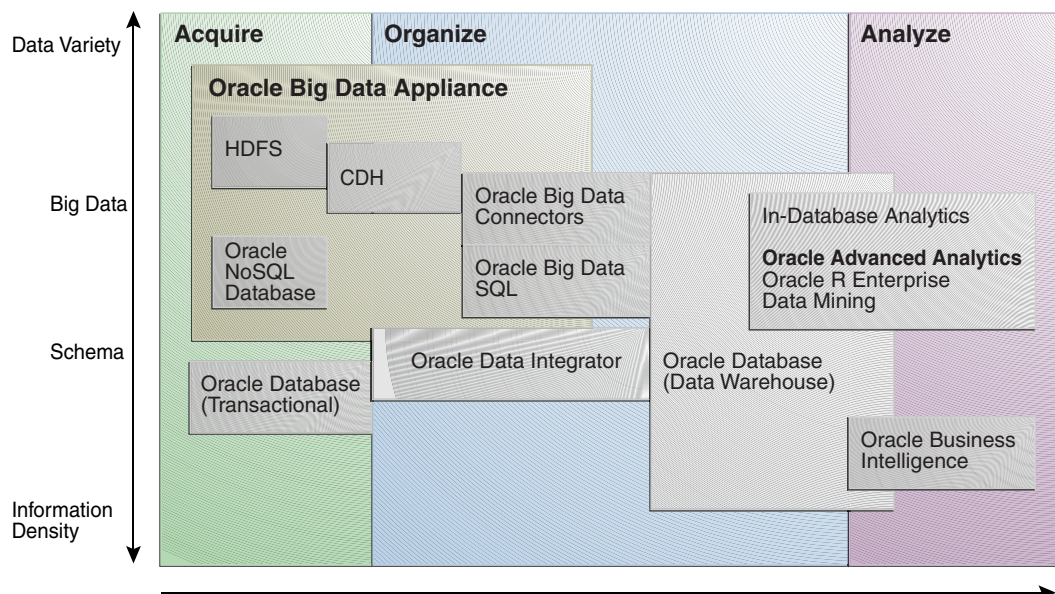
Software Component Overview

The major software components perform three basic tasks:

- Acquire
- Organize
- Analyze and visualize

The best tool for each task depends on the density of the information and the degree of structure. [Figure 1–2](#) shows the relationships among the tools and identifies the tasks that they perform.

Figure 1–2 Oracle Big Data Appliance Software Overview



Acquiring Data for Analysis

Databases used for online transaction processing (OLTP) are the traditional data sources for data warehouses. The Oracle solution enables you to analyze traditional data stores with big data in the same Oracle data warehouse. Relational data continues to be an important source of business intelligence, although it runs on separate hardware from Oracle Big Data Appliance.

Oracle Big Data Appliance provides these facilities for capturing and storing big data:

- **Hadoop Distributed File System**
- **Apache Hive**
- **Oracle NoSQL Database**

Hadoop Distributed File System

Cloudera's Distribution including Apache Hadoop (CDH) on Oracle Big Data Appliance uses the Hadoop Distributed File System (HDFS). HDFS stores extremely large files containing record-oriented data. On Oracle Big Data Appliance, HDFS splits large data files into chunks of 256 megabytes (MB), and replicates each chunk across three different nodes in the cluster. The size of the chunks and the number of replications are configurable.

Chunking enables HDFS to store files that are larger than the physical storage of one server. It also allows the data to be processed in parallel across multiple computers with multiple processors, all working on data that is stored locally. Replication ensures the high availability of the data: if a server fails, the other servers automatically take over its work load.

HDFS is typically used to store all types of big data.

See Also:

- For conceptual information about Hadoop technologies, refer to this third-party publication:
Hadoop: The Definitive Guide, Third Edition by Tom White (O'Reilly Media Inc., 2012, ISBN: 978-1449311520).
- For documentation about Cloudera's Distribution including Apache Hadoop, see the Cloudera library at <http://oracle.cloudera.com/>

Apache Hive

Hive is an open-source data warehouse that supports data summarization, ad hoc querying, and data analysis of data stored in HDFS. It uses a SQL-like language called **HiveQL**. An interpreter generates MapReduce code from the HiveQL queries. By storing data in Hive, you can avoid writing MapReduce programs in Java.

Hive is a component of CDH and is always installed on Oracle Big Data Appliance. Oracle Big Data Connectors can access Hive tables.

Oracle NoSQL Database

Oracle NoSQL Database is a distributed key-value database built on the proven storage technology of Berkeley DB Java Edition. Whereas HDFS stores unstructured data in very large files, Oracle NoSQL Database indexes the data and supports

transactions. But unlike Oracle Database, which stores highly structured data, Oracle NoSQL Database has relaxed consistency rules, no schema structure, and only modest support for joins, particularly across storage nodes.

NoSQL databases, or "Not Only SQL" databases, have developed over the past decade specifically for storing big data. However, they vary widely in implementation. Oracle NoSQL Database has these characteristics:

- Uses a system-defined, consistent hash index for data distribution
- Supports high availability through replication
- Provides single-record, single-operation transactions with relaxed consistency guarantees
- Provides a Java API

Oracle NoSQL Database is designed to provide highly reliable, scalable, predictable, and available data storage. The key-value pairs are stored in shards or partitions (that is, subsets of data) based on a primary key. Data on each shard is replicated across multiple storage nodes to ensure high availability. Oracle NoSQL Database supports fast querying of the data, typically by key lookup.

An intelligent driver links the NoSQL database with client applications and provides access to the requested key-value on the storage node with the lowest latency.

Oracle NoSQL Database includes hashing and balancing algorithms to ensure proper data distribution and optimal load balancing, replication management components to handle storage node failure and recovery, and an easy-to-use administrative interface to monitor the state of the database.

Oracle NoSQL Database is typically used to store customer profiles and similar data for identifying and analyzing big data. For example, you might log in to a website and see advertisements based on your stored customer profile (a record in Oracle NoSQL Database) and your recent activity on the site (web logs currently streaming into HDFS).

Oracle NoSQL Database is an optional component of Oracle Big Data Appliance and runs on a separate cluster from CDH.

See Also:

- *Oracle NoSQL Database Getting Started Guide* at
<http://docs.oracle.com/cd/NOSQL/html/index.html>
- *Oracle Big Data Appliance Licensing Information*

Organizing Big Data

Oracle Big Data Appliance provides several ways of organizing, transforming, and reducing big data for analysis:

- [MapReduce](#)
- [Oracle Big Data SQL](#)
- [Oracle Big Data Connectors](#)
- [Oracle R Support for Big Data](#)

MapReduce

The MapReduce engine provides a platform for the massively parallel execution of algorithms written in Java. MapReduce uses a parallel programming model for processing data on a distributed system. It can process vast amounts of data quickly and can scale linearly. It is particularly effective as a mechanism for batch processing of unstructured and semistructured data. MapReduce abstracts lower-level operations into computations over a set of keys and values.

Although big data is often described as unstructured, incoming data always has some structure. However, it does not have a fixed, predefined structure when written to HDFS. Instead, MapReduce creates the desired structure as it reads the data for a particular job. The same data can have many different structures imposed by different MapReduce jobs.

A simplified description of a MapReduce job is the successive alternation of two phases: the Map phase and the Reduce phase. Each Map phase applies a transform function over each record in the input data to produce a set of records expressed as key-value pairs. The output from the Map phase is input to the Reduce phase. In the Reduce phase, the Map output records are sorted into key-value sets, so that all records in a set have the same key value. A reducer function is applied to all the records in a set, and a set of output records is produced as key-value pairs. The Map phase is logically run in parallel over each record, whereas the Reduce phase is run in parallel over all key values.

Note: Oracle Big Data Appliance uses the Yet Another Resource Negotiator (YARN) implementation of MapReduce.

Oracle Big Data SQL

Oracle Big Data SQL supports queries against vast amounts of big data stored in multiple data sources, including Apache Hive, HDFS, Oracle NoSQL Database, and Apache HBase. You can view and analyze data from various data stores together, as if it were all stored in an Oracle database.

Using Oracle Big Data SQL, you can query data stored in a Hadoop cluster using the complete SQL syntax. You can execute the most complex SQL `SELECT` statements against data in Hadoop, either manually or using your existing applications, to tease out the most significant insights.

Oracle Big Data SQL is licensed separately from Oracle Big Data Appliance.

See Also: [Part III, "Oracle Big Data SQL"](#)

Oracle Big Data Connectors

Oracle Big Data Connectors facilitate data access between data stored in CDH and Oracle Database. The connectors are licensed separately from Oracle Big Data Appliance and include:

- [Oracle SQL Connector for Hadoop Distributed File System](#)
- [Oracle Loader for Hadoop](#)
- [Oracle XQuery for Hadoop](#)
- [Oracle R Advanced Analytics for Hadoop](#)
- [Oracle Data Integrator](#)

See Also: *Oracle Big Data Connectors User's Guide*

Oracle SQL Connector for Hadoop Distributed File System

Oracle SQL Connector for Hadoop Distributed File System (Oracle SQL Connector for HDFS) provides read access to HDFS from an Oracle database using **external tables**.

An external table is an Oracle Database object that identifies the location of data outside of the database. Oracle Database accesses the data by using the metadata provided when the external table was created. By querying the external tables, users can access data stored in HDFS as if that data were stored in tables in the database. External tables are often used to stage data to be transformed during a database load.

You can use Oracle SQL Connector for HDFS to:

- Access data stored in HDFS files
- Access Hive tables.
- Access Data Pump files generated by Oracle Loader for Hadoop
- Load data extracted and transformed by Oracle Data Integrator

Oracle Loader for Hadoop

Oracle Loader for Hadoop is an efficient and high-performance loader for fast movement of data from a Hadoop cluster into a table in an Oracle database. It can read and load data from a wide variety of formats. Oracle Loader for Hadoop partitions the data and transforms it into a database-ready format in Hadoop. It optionally sorts records by primary key before loading the data or creating output files. The load runs as a MapReduce job on the Hadoop cluster.

Oracle Data Integrator

Oracle Data Integrator (ODI) extracts, transforms, and loads data into Oracle Database from a wide range of sources.

In ODI, a knowledge module (KM) is a code template dedicated to a specific task in the data integration process. You use Oracle Data Integrator Studio to load, select, and configure the KMs for your particular application. More than 150 KMs are available to help you acquire data from a wide range of third-party databases and other data repositories. You only need to load a few KMs for any particular job.

Oracle Data Integrator contains the KMs specifically for use with big data.

Oracle XQuery for Hadoop

Oracle XQuery for Hadoop runs transformations expressed in the XQuery language by translating them into a series of MapReduce jobs, which are executed in parallel on the Hadoop cluster. The input data can be located in HDFS or Oracle NoSQL Database. Oracle XQuery for Hadoop can write the transformation results to HDFS, Oracle NoSQL Database, or Oracle Database.

Oracle R Advanced Analytics for Hadoop

Oracle R Advanced Analytics for Hadoop is a collection of R packages that provides:

- Interfaces to work with Hive tables, Apache Hadoop compute infrastructure, local R environment and database tables
- Predictive analytic techniques written in R or Java as Hadoop MapReduce jobs that can be applied to data in HDFS files

Using simple R functions, you can copy data between R memory, the local file system, HDFS, and Hive. You can write mappers and reducers in R, schedule these R programs to execute as Hadoop MapReduce jobs, and return the results to any of those locations.

Oracle R Support for Big Data

R is an open-source language and environment for statistical analysis and graphing. It provides linear and nonlinear modeling, standard statistical methods, time-series analysis, classification, clustering, and graphical data displays. Thousands of open-source packages are available in the Comprehensive R Archive Network (CRAN) for a spectrum of applications, such as bioinformatics, spatial statistics, and financial and marketing analysis. The popularity of R has increased as its functionality matured to rival that of costly proprietary statistical packages.

Analysts typically use R on a PC, which limits the amount of data and the processing power available for analysis. Oracle eliminates this restriction by extending the R platform to directly leverage Oracle Big Data Appliance. Oracle R Distribution is installed on all nodes of Oracle Big Data Appliance.

Oracle R Advanced Analytics for Hadoop provides R users with high-performance, native access to HDFS and the MapReduce programming framework, which enables R programs to run as MapReduce jobs on vast amounts of data. Oracle R Advanced Analytics for Hadoop is included in the Oracle Big Data Connectors. See "[Oracle R Advanced Analytics for Hadoop](#)" on page 1-8.

Oracle R Enterprise is a component of the Oracle Advanced Analytics option to Oracle Database. It provides:

- Transparent access to database data for data preparation and statistical analysis from R
- Execution of R scripts at the database server, accessible from both R and SQL
- A wide range of predictive and data mining in-database algorithms

Oracle R Enterprise enables you to store the results of your analysis of big data in an Oracle database, or accessed for display in dashboards and applications.

Both Oracle R Advanced Analytics for Hadoop and Oracle R Enterprise make Oracle Database and the Hadoop computational infrastructure available to statistical users without requiring them to learn the native programming languages of either one.

See Also:

- For information about R, go to <http://www.r-project.org/>
- For information about Oracle R Enterprise, go to http://docs.oracle.com/cd/E40980_01/welcome.html

Analyzing and Visualizing Big Data

After big data is transformed and loaded in Oracle Database, you can use the full spectrum of Oracle business intelligence solutions and decision support products to further analyze and visualize all your data.

See Also:

- Oracle Business Intelligence website at
<http://www.oracle.com/us/solutions/ent-performance-bi/business-intelligence/index.html>
- Data Warehousing and Business Intelligence in the Oracle Database Documentation Library at
http://www.oracle.com/pls/db112/portal.portal_db?selected=6&frame=

Administering Oracle Big Data Appliance

This chapter provides information about the software and services installed on Oracle Big Data Appliance. It contains these sections:

- [Monitoring Multiple Clusters Using Oracle Enterprise Manager](#)
- [Managing Operations Using Cloudera Manager](#)
- [Using Hadoop Monitoring Utilities](#)
- [Using Cloudera Hue to Interact With Hadoop](#)
- [About the Oracle Big Data Appliance Software](#)
- [About the CDH Software Services](#)
- [Effects of Hardware on Software Availability](#)
- [Managing a Hardware Failure](#)
- [Stopping and Starting Oracle Big Data Appliance](#)
- [Managing Oracle Big Data SQL](#)
- [Security on Oracle Big Data Appliance](#)
- [Auditing Oracle Big Data Appliance](#)
- [Collecting Diagnostic Information for Oracle Customer Support](#)

Monitoring Multiple Clusters Using Oracle Enterprise Manager

An Oracle Enterprise Manager plug-in enables you to use the same system monitoring tool for Oracle Big Data Appliance as you use for Oracle Exadata Database Machine or any other Oracle Database installation. With the plug-in, you can view the status of the installed software components in tabular or graphic presentations, and start and stop these software services. You can also monitor the health of the network and the rack components.

Oracle Enterprise Manager enables you to monitor all Oracle Big Data Appliance racks on the same InfiniBand fabric. It provides summary views of both the rack hardware and the software layout of the logical clusters.

Using the Enterprise Manager Web Interface

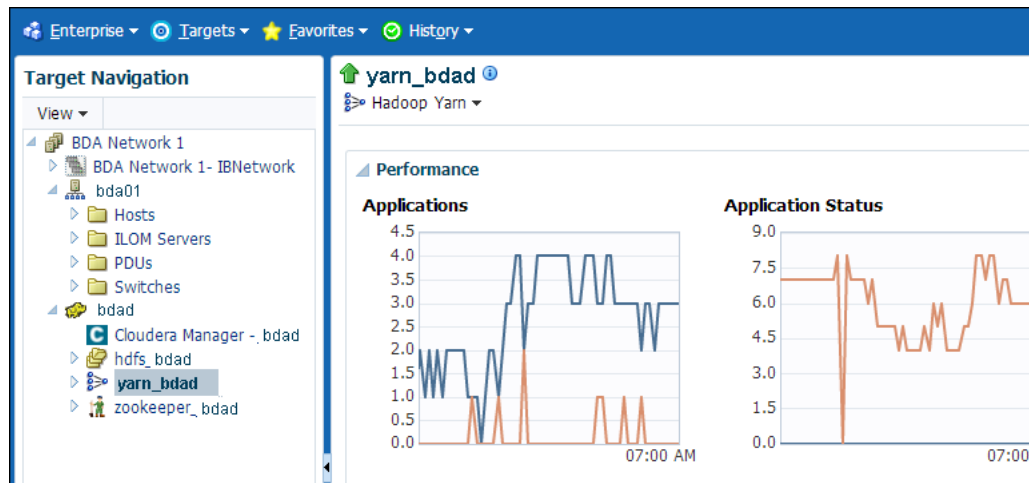
After opening Oracle Enterprise Manager web interface, logging in, and selecting a target cluster, you can drill down into these primary areas:

- **InfiniBand network:** Network topology and status for InfiniBand switches and ports. See [Figure 2-1](#).

- **Hadoop cluster:** Software services for HDFS, MapReduce, and ZooKeeper.
- **Oracle Big Data Appliance rack:** Hardware status including server hosts, Oracle Integrated Lights Out Manager (Oracle ILOM) servers, power distribution units (PDUs), and the Ethernet switch.

Figure 2–1 shows a small section of the cluster home page.

Figure 2–1 YARN Page in Oracle Enterprise Manager



To monitor Oracle Big Data Appliance using Oracle Enterprise Manager:

1. Download and install the plug-in. See *Oracle Enterprise Manager System Monitoring Plug-in Installation Guide for Oracle Big Data Appliance*.
2. Log in to Oracle Enterprise Manager as a privileged user.
3. From the Targets menu, choose **Big Data Appliance** to view the Big Data page. You can see the overall status of the targets already discovered by Oracle Enterprise Manager.
4. Select a target cluster to view its detail pages.
5. Expand the target navigation tree to display the components. Information is available at all levels.
6. Select a component in the tree to display its home page.
7. To change the display, choose an item from the drop-down menu at the top left of the main display area.

See Also: *Oracle Enterprise Manager System Monitoring Plug-in Installation Guide for Oracle Big Data Appliance* for installation instructions and use cases.

Using the Enterprise Manager Command-Line Interface

The Enterprise Manager command-line interface (`emcli`) is installed on Oracle Big Data Appliance along with all the other software. It provides the same functionality as the web interface. You must provide credentials to connect to Oracle Management Server.

To get help, enter `emcli help`.

See Also: *Oracle Enterprise Manager Command Line Interface Guide*

Managing Operations Using Cloudera Manager

Cloudera Manager is installed on Oracle Big Data Appliance to help you with Cloudera's Distribution including Apache Hadoop (CDH) operations. Cloudera Manager provides a single administrative interface to all Oracle Big Data Appliance servers configured as part of the Hadoop cluster.

Cloudera Manager simplifies the performance of these administrative tasks:

- Monitor jobs and services
- Start and stop services
- Manage security and Kerberos credentials
- Monitor user activity
- Monitor the health of the system
- Monitor performance metrics
- Track hardware use (disk, CPU, and RAM)

Cloudera Manager runs on the ResourceManager node (node03) and is available on port 7180.

To use Cloudera Manager:

1. Open a browser and enter a URL like the following:

```
http://bda1node03.example.com:7180
```

In this example, bda1 is the name of the appliance, node03 is the name of the server, example.com is the domain, and 7180 is the default port number for Cloudera Manager.

2. Log in with a user name and password for Cloudera Manager. Only a user with administrative privileges can change the settings. Other Cloudera Manager users can view the status of Oracle Big Data Appliance.

See Also: *Cloudera Manager Monitoring and Diagnostics Guide* at

<http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM5/latest/Cloudera-Manager-Diagnostics-Guide/Cloudera-Manager-Diagnostics-Guide.html>

or click **Help** on the Cloudera Manager Support menu

Monitoring the Status of Oracle Big Data Appliance

In Cloudera Manager, you can choose any of the following pages from the menu bar across the top of the display:

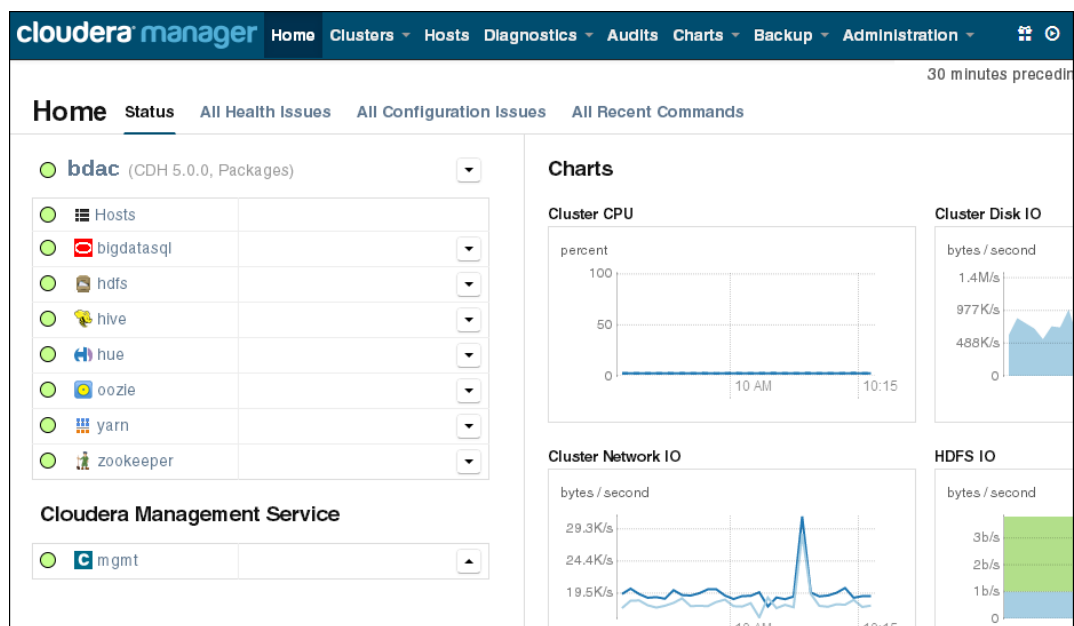
- **Home:** Provides a graphic overview of activities and links to all services controlled by Cloudera Manager. See [Figure 2–2](#).
- **Clusters:** Accesses the services on multiple clusters.
- **Hosts:** Monitors the health, disk usage, load, physical memory, swap space, and other statistics for all servers in the cluster.
- **Diagnostics:** Accesses events and logs. Cloudera Manager collects historical information about the systems and services. You can search for a particular phrase

for a selected server, service, and time period. You can also select the minimum severity level of the logged messages included in the search: TRACE, DEBUG, INFO, WARN, ERROR, or FATAL.

- **Audits:** Displays the audit history log for a selected time range. You can filter the results by user name, service, or other criteria, and download the log as a CSV file.
- **Charts:** Enables you to view metrics from the Cloudera Manager time-series data store in a variety of chart types, such as line and bar.
- **Backup:** Accesses snapshot policies and scheduled replications.
- **Administration:** Provides a variety of administrative options, including Settings, Alerts, Users, and Kerberos.

Figure 2–2 shows the Cloudera Manager home page.

Figure 2–2 Cloudera Manager Home Page



Performing Administrative Tasks

As a Cloudera Manager administrator, you can change various properties for monitoring the health and use of Oracle Big Data Appliance, add users, and set up Kerberos security.

To access Cloudera Manager Administration:

1. Log in to Cloudera Manager with administrative privileges.
2. Click **Administration**, and select a task from the menu.

Managing CDH Services With Cloudera Manager

Cloudera Manager provides the interface for managing these services:

- HDFS
- Hive
- Hue

- Oozie
- YARN
- ZooKeeper

You can use Cloudera Manager to change the configuration of these services, stop, and restart them. Additional services are also available, which require configuration before you can use them. See ["Unconfigured Software"](#) on page 2-9.

Note: Manual edits to Linux service scripts or Hadoop configuration files do not affect these services. You must manage and configure them using Cloudera Manager.

Using Hadoop Monitoring Utilities

You also have the option of using the native Hadoop utilities. These utilities are read-only and do not require authentication.

Cloudera Manager provides an easy way to obtain the correct URLs for these utilities. On the YARN service page, expand the Web UI submenu.

Monitoring MapReduce Jobs

You can monitor MapReduce jobs using the resource manager interface.

To monitor MapReduce jobs:

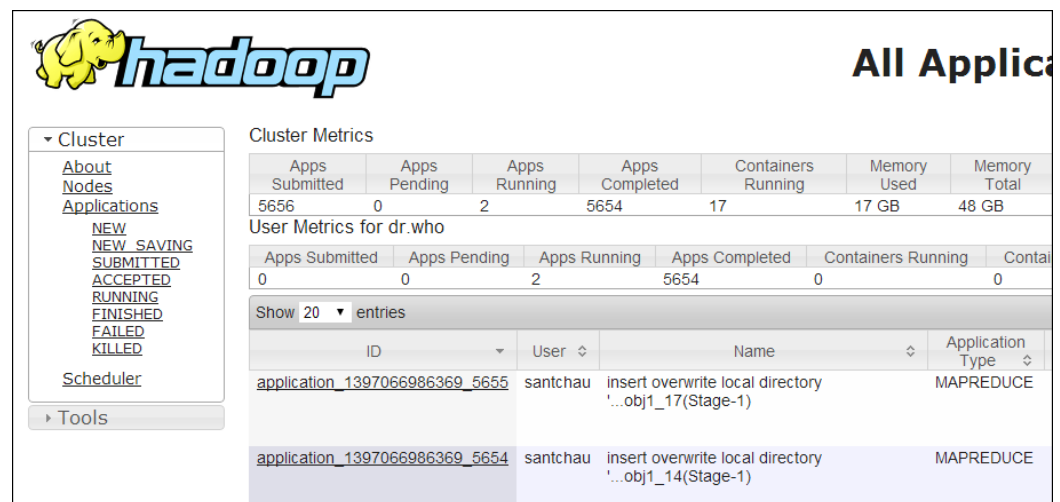
- Open a browser and enter a URL like the following:

`http://bda1node03.example.com:8088`

In this example, bda1 is the name of the rack, node03 is the name of the server where the YARN resource manager runs, and 8088 is the default port number for the user interface.

[Figure 2-3](#) shows the resource manager interface.

Figure 2-3 YARN Resource Manager Interface



Monitoring the Health of HDFS

You can monitor the health of the Hadoop file system by using the DFS health utility on the first two nodes of a cluster.

To monitor HDFS:

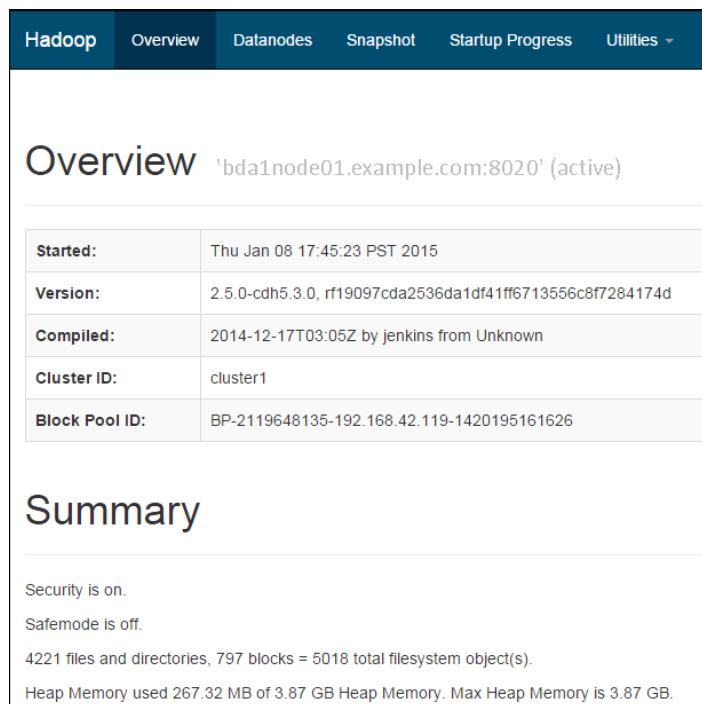
- Open a browser and enter a URL like the following:

`http://bda1node01.example.com:50070`

In this example, `bda1` is the name of the rack, `node01` is the name of the server where the `dfshealth` utility runs, and `50070` is the default port number for the user interface.

Figure 2–3 shows the DFS health utility interface.

Figure 2–4 DFS Health Utility



Using Cloudera Hue to Interact With Hadoop

Hue runs in a browser and provides an easy-to-use interface to several applications to support interaction with Hadoop and HDFS. You can use Hue to perform any of the following tasks:

- Query Hive data stores
- Create, load, and delete Hive tables
- Work with HDFS files and directories
- Create, submit, and monitor MapReduce jobs
- Monitor MapReduce jobs
- Create, edit, and submit workflows using the Oozie dashboard
- Manage users and groups

Hue is automatically installed and configured on Oracle Big Data Appliance. It runs on port 8888 of the ResourceManager node (node03).

To use Hue:

1. Log in to Cloudera Manager and click the **hue** service on the Home page.
2. On the hue page under Quick Links, click Hue Web UI.
3. Bookmark the Hue URL, so that you can open Hue directly in your browser. The following URL is an example:

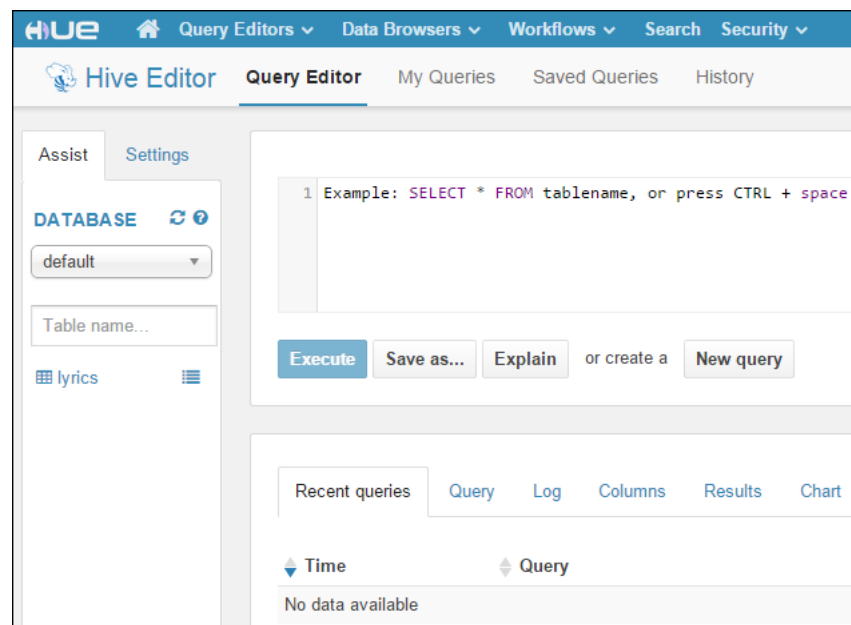
`http://bdainode03.example.com:8888`

4. Log in with your Hue credentials.

Oracle Big Data Appliance is not configured initially with any Hue user accounts. The first user who connects to Hue can log in with any user name and password, and automatically becomes an administrator. This user can create other user and administrator accounts.

Figure 2–5 shows the Hive Query Editor.

Figure 2–5 *Hive Query Editor*



See Also: *Hue User Guide* at

<http://archive-primary.cloudera.com/cdh5/cdh/5/hue/user-guide/>

About the Oracle Big Data Appliance Software

The following sections identify the software installed on Oracle Big Data Appliance. Some components operate with Oracle Database 11.2.0.2 and later releases.

This section contains the following topics:

- [Software Components](#)
- [Unconfigured Software](#)

- [Allocating Resources Among Services](#)

Software Components

These software components are installed on all servers in the cluster. Oracle Linux, required drivers, firmware, and hardware verification utilities are factory installed. All other software is installed on site. The optional software components may not be configured in your installation.

Note: You do not need to install additional software on Oracle Big Data Appliance. Doing so may result in a loss of warranty and support. See the *Oracle Big Data Appliance Owner's Guide*.

Base image software:

- Oracle Linux 6.5 (upgrades stay at 5.8) with Oracle Unbreakable Enterprise Kernel version 2 (UEK2)
- Java [HotSpot](#) Virtual Machine 7 version 72 (JDK 7u72)
- [Oracle R Distribution](#) 3.1.1-2
- [MySQL Database](#) 5.5.35 Advanced Edition
- Puppet, firmware, Oracle Big Data Appliance utilities
- Oracle InfiniBand software

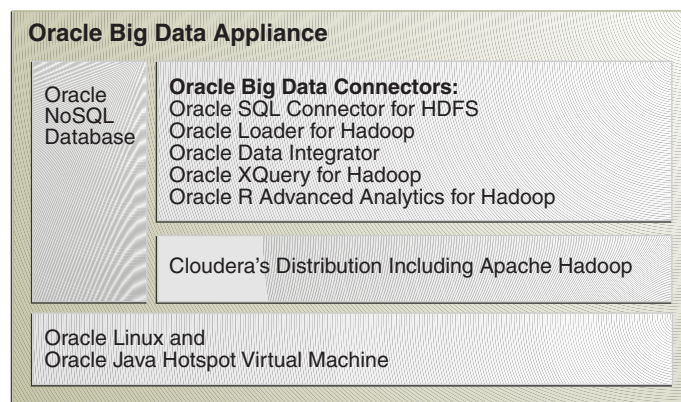
Mammoth installation:

- Cloudera's Distribution including Apache Hadoop Release 5 (5.3.0) including:
 - Apache Hive
 - Apache HBase
 - [Apache Sentry](#)
 - Apache Spark
 - [Cloudera Impala](#)
 - [Cloudera Search](#)
- Cloudera Manager Release 5 (5.3.0) including [Cloudera Navigator](#)
- [Oracle Database Instant Client](#) 12.1
- Oracle Big Data SQL (optional)
- Oracle NoSQL Database Community Edition or Enterprise Edition 12c Release 1 Version 3.2.4 (optional)
- Oracle Big Data Connectors 4.1 (optional):
 - Oracle SQL Connector for Hadoop Distributed File System (HDFS)
 - Oracle Loader for Hadoop
 - Oracle Data Integrator Agent 12.1.3.0
 - Oracle XQuery for Hadoop
 - Oracle R Advanced Analytics for Hadoop

See Also: *Oracle Big Data Appliance Owner's Guide* for information about the Mammoth utility

Figure 2–6 shows the relationships among the major components.

Figure 2–6 Major Software Components of Oracle Big Data Appliance



Unconfigured Software

Your Oracle Big Data Appliance license includes all components in Cloudera Enterprise Data Hub Edition. All CDH components are installed automatically by the Mammoth utility. Do not download them from the Cloudera website.

However, you must use Cloudera Manager to add some services before you can use them, such as the following:

- [Apache Flume](#)
- [Apache HBase](#)
- [Apache Spark](#)
- [Apache Sqoop](#)
- [Cloudera Impala](#)
- [Cloudera Navigator](#)
- [Cloudera Search](#)

To add a service:

1. Log in to Cloudera Manager as the admin user.
2. On the Home page, expand the cluster menu in the left panel and choose **Add a Service** to open the Add Service wizard. The first page lists the services you can add.
3. Follow the steps of the wizard.

See Also:

- For a complete list of CDH components:
<http://www.cloudera.com/content/cloudera/en/products-and-services/product-comparison.html>
- *CDH5 Installation and Configuration Guide* for configuration procedures at
<http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Installation-Guide/CDH5-Installation-Guide.html>

Allocating Resources Among Services

You can allocate resources to each service—HDFS, YARN, Oracle Big Data SQL, Hive, and so forth—as a percentage of the total resource pool. Cloudera Manager automatically calculates the recommended resource management settings based on these percentages. The static service pools isolate services on the cluster, so that a high load on one service has a limited impact on the other services.

To allocate resources among services:

1. Log in as `admin` to Cloudera Manager.
2. Open the Clusters menu at the top of the page, then select **Static Service Pools** under Resource Management.
3. Select **Configuration**.
4. Follow the steps of the wizard, or click **Change Settings Directly** to edit the current settings.

About the CDH Software Services

All services are installed on all nodes in a CDH cluster, but individual services run only on designated nodes. There are slight variations in the location of the services depending on the configuration of the cluster.

This section describes the services in a default YARN configuration.

This section contains the following topics:

- [Where Do the Services Run on a Single-Rack CDH Cluster?](#)
- [Where Do the Services Run on a Multirack CDH Cluster?](#)
- [About MapReduce](#)
- [Automatic Failover of the NameNode](#)
- [Automatic Failover of the ResourceManager](#)

Where Do the Services Run on a Single-Rack CDH Cluster?

[Table 2–1](#) identifies the services in CDH clusters configured within a single rack, including starter racks and clusters with six nodes. Node01 is the first server in the cluster (server 1, 7, or 10), and `nodemn` is the last server in the cluster (server 6, 9, 12, or 18).

Table 2–1 Service Locations for One or More CDH Clusters in a Single Rack

Node01	Node02	Node03	Node04	Node05 to <i>nn</i>
Balancer		Cloudera Manager Server		
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
DataNode	DataNode	DataNode	DataNode	DataNode
Failover Controller	Failover Controller	JobHistory	Hive, Hue, Oozie, Solr	
JournalNode	JournalNode	JournalNode		
	MySQL Backup	MySQL Primary		
NameNode	NameNode			
NodeManager ¹	NodeManager ¹	NodeManager	NodeManager	NodeManager
			Oracle Data Integrator Agent	
Puppet	Puppet	Puppet	Puppet	Puppet
Puppet Master		ResourceManager	ResourceManager	
ZooKeeper	ZooKeeper	ZooKeeper		

¹ Starter racks and six-node clusters only, with reduced allocated resources

Where Do the Services Run on a Multirack CDH Cluster?

When multiple racks are configured as a single CDH cluster, some critical services are installed on the second rack. The second rack must have at least two nodes.

Table 2–2 identifies the location of services on the first rack of a multirack cluster.

Table 2–2 Service Locations in the First Rack of a Multirack CDH Cluster

Node01	Node02	Node03	Node04	Node05 to <i>nn</i> ¹
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
		Cloudera Manager Server		
DataNode	DataNode	DataNode	DataNode	DataNode
Failover Controller				
JournalNode	JournalNode			
NameNode	MySQL Primary			
NodeManager	NodeManager	NodeManager	NodeManager	NodeManager
Puppet	Puppet	Puppet	Puppet	Puppet
Puppet Master	ResourceManager			
ZooKeeper	ZooKeeper			

¹ *nn* includes the servers in additional racks.

Table 2–3 shows the service locations in the second rack of a multirack cluster.

Table 2–3 Service Locations in the Second Rack of a Multirack CDH Cluster

Node01	Node02	Node03	Node04	Node05 to nn
Balancer				
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
DataNode	DataNode	DataNode	DataNode	DataNode
Failover Controller				
JournalNode	Hive, Hue, Oozie, Solr			
MySQL Backup				
NameNode				
NodeManager ¹	NodeManager ¹	NodeManager	NodeManager	NodeManager
	Oracle Data Integrator Agent			
Puppet	Puppet	Puppet	Puppet	Puppet
Puppet Master	ResourceManager			
ZooKeeper				

¹ Starter racks and six-node clusters only, with reduced allocated resources

Note: When expanding a cluster from one to two racks, Mammoth moves all critical services from nodes 2 and 4 of the first rack to nodes 1 and 2 of the second rack. Node 2 of the first rack becomes a noncritical node.

About MapReduce

Yet Another Resource Negotiator (YARN) is the version of MapReduce that runs on Oracle Big Data Appliance, beginning with version 3.0. MapReduce applications developed using MapReduce 1 (MRv1) may require recompilation to run under YARN.

The ResourceManager performs all resource management tasks. An MRAppMaster performs the job management tasks. Each job has its own MRAppMaster. The NodeManager has containers that can run a map task, a reduce task, or an MRAppMaster. The NodeManager can dynamically allocate containers using the available memory. This architecture results in improved scalability and better use of the cluster than MRv1.

YARN also manages resources for Spark and Impala.

See Also: "Running Existing Applications on Hadoop 2 YARN" at

<http://hortonworks.com/blog/running-existing-applications-on-hadoop-2-yarn/>

Automatic Failover of the NameNode

The NameNode is the most critical process because it keeps track of the location of all data. Without a healthy NameNode, the entire cluster fails. Apache Hadoop v0.20.2 and earlier are vulnerable to failure because they have a single name node.

Cloudera's Distribution including Apache Hadoop Version 4 (CDH5) reduces this vulnerability by maintaining redundant NameNodes. The data is replicated during normal operation as follows:

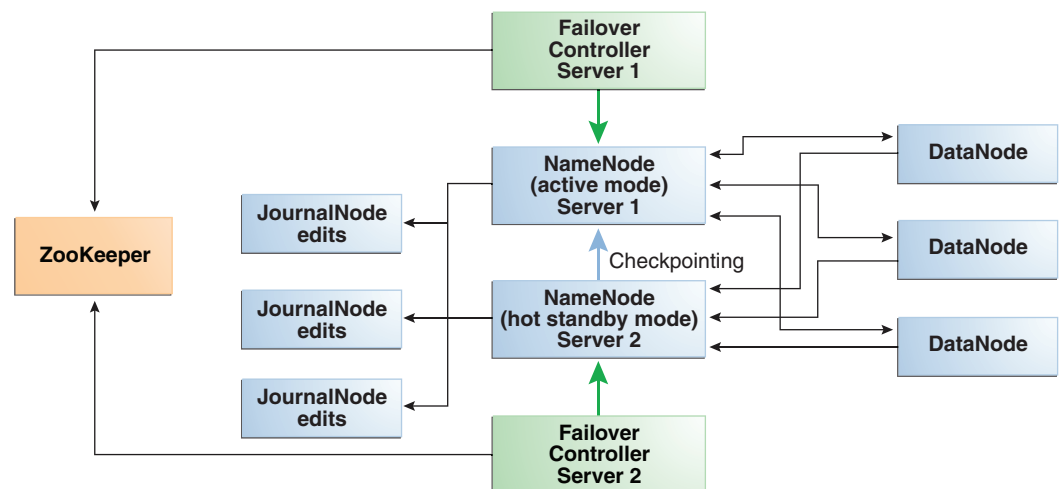
- CDH maintains redundant NameNodes on the first two nodes of a cluster. One of the NameNodes is in active mode, and the other NameNode is in hot standby mode. If the active NameNode fails, then the role of active NameNode automatically fails over to the standby NameNode.
- The NameNode data is written to a mirrored partition so that the loss of a single disk can be tolerated. This mirroring is done at the factory as part of the operating system installation.
- The active NameNode records all changes to the file system metadata in at least two JournalNode processes, which the standby NameNode reads. There are three JournalNodes, which run on the first three nodes of each cluster.
- The changes recorded in the journals are periodically consolidated into a single fsimage file in a process called **checkpointing**.

On Oracle Big Data Appliance, the default log level of the NameNode is `DEBUG`, to support the Oracle Audit Vault and Database Firewall plugin. If this option is not configured, then you can reset the log level to `INFO`.

Note: Oracle Big Data Appliance 2.0 and later releases do not support the use of an external NFS filer for backups and do not use NameNode federation.

Figure 2–7 shows the relationships among the processes that support automatic failover of the NameNode.

Figure 2–7 Automatic Failover of the NameNode on Oracle Big Data Appliance



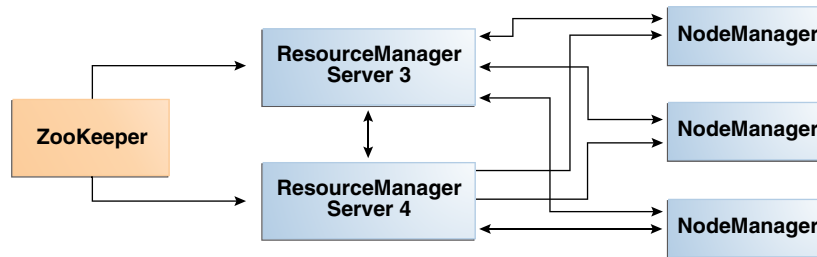
Automatic Failover of the ResourceManager

The ResourceManager allocates resources for application tasks and application masters across the cluster. Like the NameNode, the ResourceManager is a critical point of failure for the cluster. If all ResourceManagers fail, then all jobs stop running. Oracle Big Data Appliance 3.0 supports ResourceManager High Availability in Cloudera 5 to reduce this vulnerability.

CDH maintains redundant ResourceManager services on node03 and node04. One of the services is in active mode, and the other service is in hot standby mode. If the active service fails, then the role of active ResourceManager automatically fails over to the standby service. No failover controllers are required.

Figure 2–7 shows the relationships among the processes that support automatic failover of the ResourceManager.

Figure 2–8 Automatic Failover of the ResourceManager on Oracle Big Data Appliance



Map and Reduce Resource Configuration

The NodeManager services are allocated a fixed amount of memory and virtual core (Vcore), as shown in Table 2–4.

Table 2–4 Resource Allocation for NodeManagers

Appliance Model	Memory	VCore
Oracle Big Data Appliance X4-2	40 GB	32
Oracle Big Data Appliance X3-2	40 GB	32
Oracle Big Data Appliance (Sun Fire X4270 M2-based rack)	30 GB	24

Each node in a cluster has a maximum number of map and reduce tasks that are allowed to run simultaneously. Table 2–5 shows the default configuration of resources for the MapReduce service on Oracle Big Data Appliance X4-2 and X3-2.

Table 2–5 Maximum Map and Reduce Tasks on Oracle Big Data Appliance X3-2

Node	6-Node Cluster	Larger Clusters ¹
Node01 and Node02	14 map 8 reduce	None (no NodeManager)
Node03 and Node04	10 map 6 reduce	10 map 6 reduce
Node05 to Nodenn	20 map 13 reduce	20 map 13 reduce

¹ 9 or more nodes

Table 2–6 shows the default configuration of resources for the MapReduce service on Oracle Big Data Appliance Sun Fire X4270 M2-based racks.

Table 2–6 Maximum Map and Reduce Tasks on Sun Fire X4270 M2-Based Racks

Node	6-Node Cluster	Larger Clusters ¹
Node01 and Node02	10 map 6 reduce	None
Node03 and Node04	7 map 4 reduce	7 map 4 reduce
Node05 to Nodenn	15 map 10 reduce	15 map 10 reduce

¹ 9 or more nodes

Effects of Hardware on Software Availability

The effects of a server failure vary depending on the server's function within the CDH cluster. Oracle Big Data Appliance servers are more robust than commodity hardware, so you should experience fewer hardware failures. This section highlights the most important services that run on the various servers of the primary rack. For a full list, see ["Where Do the Services Run on a Single-Rack CDH Cluster?"](#) on page 2-10.

Note: In a multirack cluster, some critical services run on the first server of the second rack. See ["Where Do the Services Run on a Multirack CDH Cluster?"](#) on page 2-11.

Logical Disk Layout

Each server has 12 disks. The critical operating system is stored on disks 1 and 2.

[Table 2–7](#) describes how the disks are partitioned.

Table 2–7 Logical Disk Layout

Disk	Description
1 to 2	150 gigabytes (GB) physical and logical partition, mirrored to create two copies, with the Linux operating system, all installed software, NameNode data, and MySQL Database data. The NameNode and MySQL Database data are replicated on two servers for a total of four copies. 2.8 terabytes (TB) HDFS data partition
3 to 12	Single HDFS or Oracle NoSQL Database data partition

Critical and Noncritical CDH Nodes

Critical nodes are required for the cluster to operate normally and provide all services to users. In contrast, the cluster continues to operate with no loss of service when a noncritical node fails.

On single-rack clusters, the critical services are installed initially on the first four nodes of the cluster. The remaining nodes (node05 up to node18) only run noncritical services. If a hardware failure occurs on one of the critical nodes, then the services can be moved to another, noncritical server. For example, if node02 fails, then you might move its critical services node05. [Table 2–1](#) identifies the initial location of services for clusters that are configured on a single rack.

In a multirack cluster, some critical services run on the first server of the second rack. See ["Where Do the Services Run on a Single-Rack CDH Cluster?"](#) on page 2-10.

High Availability or Single Points of Failure?

Some services have high availability and automatic failover. Other services have a single point of failure. The following list summarizes the critical services:

- **NameNodes:** High availability with automatic failover
- **ResourceManagers:** High availability with automatic failover
- **MySQL Database:** Primary and backup databases are configured with replication of the primary database to the backup database. There is no automatic failover. If the primary database fails, the functionality of the cluster is diminished, but no data is lost.
- **Cloudera Manager:** The Cloudera Manager server runs on one node. If it fails, then Cloudera Manager functionality is unavailable.
- **Oozie server, Hive server, Hue server, and Oracle Data Integrator agent:** These services have no redundancy. If the node fails, then the services are unavailable.

Where Do the Critical Services Run?

[Table 2–8](#) identifies where the critical services run in a CDH cluster. These four nodes are described in more detail in the topics that follow.

Table 2–8 Critical Service Locations on a Single Rack

Node Name	Critical Functions
First NameNode	Balancer, Failover Controller, JournalNode, NameNode, Puppet Master, ZooKeeper
Second NameNode	Failover Controller, JournalNode, MySQL Backup Database, NameNode, ZooKeeper
First ResourceManager Node	Cloudera Manager Server, JobHistory, JournalNode, MySQL Primary Database, ResourceManager, ZooKeeper
Second ResourceManager Node	Hive, Hue, Oozie, Solr, Oracle Data Integrator Agent, ResourceManager

In a single-rack cluster, the four critical nodes are created initially on the first four nodes. See ["Where Do the Services Run on a Single-Rack CDH Cluster?"](#) on page 2-10

In a multirack cluster, the Second NameNode and the Second ResourceManager nodes are moved to the first two nodes of the second rack. See ["Where Do the Services Run on a Multirack CDH Cluster?"](#) on page 2-11.

First NameNode Node

If the first NameNode fails or goes offline (such as a restart), then the second NameNode automatically takes over to maintain the normal activities of the cluster.

Alternatively, if the second NameNode is already active, it continues without a backup. With only one NameNode, the cluster is vulnerable to failure. The cluster has lost the redundancy needed for automatic failover.

The puppet master also runs on this node. The Mammoth utility uses Puppet, and so you cannot install or reinstall the software if, for example, you must replace a disk drive elsewhere in the rack.

Second NameNode Node

If the second NameNode fails, then the function of the NameNode either fails over to the first NameNode (node01) or continues there without a backup. However, the cluster has lost the redundancy needed for automatic failover if the first NameNode also fails.

The MySQL backup database also runs on this node. MySQL Database continues to run, although there is no backup of the master database.

First ResourceManager Node

If the first ResourceManager node fails or goes offline (such as a restart), then the second ResourceManager automatically takes over to distribute MapReduce tasks to specific nodes across the cluster.

Alternatively, if the second ResourceManager is already active, it continues without a backup. With only one ResourceManager, the cluster is vulnerable to failure. The cluster has lost the redundancy needed for automatic failover.

These services are also disrupted:

- **Cloudera Manager:** This tool provides central management for the entire CDH cluster. Without this tool, you can still monitor activities using the utilities described in ["Using Hadoop Monitoring Utilities"](#) on page 2-5.
- **MySQL Database:** Cloudera Manager, Oracle Data Integrator, Hive, and Oozie use MySQL Database. The data is replicated automatically, but you cannot access it when the master database server is down.

Second ResourceManager Node

If the second ResourceManager node fails, then the function of the ResourceManager either fails over to the first ResourceManager or continues there without a backup. However, the cluster has lost the redundancy needed for automatic failover if the first ResourceManager also fails.

These services are also disrupted:

- **Oracle Data Integrator Agent** This service supports Oracle Data Integrator, which is one of the Oracle Big Data Connectors. You cannot use Oracle Data Integrator when the ResourceManager node is down.
- **Hive:** Hive provides a SQL-like interface to data that is stored in HDFS. Oracle Big Data SQL and most of the Oracle Big Data Connectors can access Hive tables, which are not available if this node fails.
- **Hue:** This administrative tool is not available when the ResourceManager node is down.
- **Oozie:** This workflow and coordination service runs on the ResourceManager node, and is unavailable when the node is down.

Noncritical CDH Nodes

The noncritical nodes are optional in that Oracle Big Data Appliance continues to operate with no loss of service if a failure occurs. The NameNode automatically replicates the lost data to always maintain three copies. MapReduce jobs execute on copies of the data stored elsewhere in the cluster. The only loss is in computational power, because there are fewer servers on which to distribute the work.

Managing a Hardware Failure

If a server starts failing, you must take steps to maintain the services of the cluster with as little interruption as possible. You can manage a failing server easily using the `bdaccli` utility, as described in the following procedures. One of the management steps is called decommissioning. *Decommissioning* stops all roles for all services, thereby preventing data loss. Cloudera Manager requires that you decommission a CDH node before retiring it.

When a noncritical node fails, there is no loss of service. However, when a critical node fails in a CDH cluster, services with a single point of failure are unavailable, as described in ["Effects of Hardware on Software Availability"](#) on page 2-15. You must decide between these alternatives:

- Wait for repairs to be made, and endure the loss of service until they are complete.
- Move the critical services to another node. This choice may require that some clients are reconfigured with the address of the new node. For example, if the second ResourceManager node (typically node03) fails, then users must redirect their browsers to the new node to access Cloudera Manager.

You must weigh the loss of services against the inconvenience of reconfiguring the clients.

About Oracle NoSQL Database Clusters

Oracle NoSQL Database clusters do not have critical nodes. The storage nodes are replicated, and users can choose from three administrative processes on different nodes. There is no loss of services.

If the node hosting Mammoth fails (the first node of the cluster), then follow the procedure for reinstalling it in ["Prerequisites for Managing a Failing Node"](#) on page 2-18

To repair or replace any failing NoSQL node, follow the procedure in ["Managing a Failing Noncritical Node"](#) on page 2-19.

Prerequisites for Managing a Failing Node

Ensure that you do the following before managing a failing or failed server, whether it is configured as a CDH node or an Oracle NoSQL Database node:

- Try restarting the services or rebooting the server.
- Determine whether the failing node is critical or noncritical.
- If the failing node is where Mammoth is installed:
 1. For a CDH node, select a noncritical node in the same cluster as the failing node.

For a NoSQL node, repair or replace the failed server first, and use it for these steps.

2. Upload the Mammoth bundle to that node and unzip it.
3. Extract all files from `BDAMammoth-version.run`, using a command like the following:

```
# ./BDAMammoth-ol6-4.0.0.run
```

Afterward, you must run all Mammoth operations from this node.

See *Oracle Big Data Appliance Owner's Guide* for information about the Mammoth utility.

4. Follow the appropriate procedure in this section for managing a failing node.

Mammoth is installed on the first node of the cluster, unless its services were migrated previously.

Managing a Failing CDH Critical Node

Only CDH clusters have critical nodes.

To manage a failing critical node:

1. Log in as root to the node where Mammoth is installed.
2. Migrate the services to a noncritical node. Replace *node_name* with the name of the failing node, such as bda1node02.

```
bdacli admin_cluster migrate node_name
```

When the command finishes, *node_name* is decommissioned and its services are running on a previously noncritical node.

3. Announce the change to the user community, so that they can redirect their clients to the new critical node as required.
4. Repair or replace the failed server.
5. From the Mammoth node as root, reprovision the repaired or replaced server as a noncritical node. Use the same name as the migrated node for *node_name*, such as bda1node02:

```
bdacli admin_cluster reprovision node_name
```

6. If the failed node supported services like HBase or Impala, which Mammoth installs but does not configure, then use Cloudera Manager to reconfigure them on the new node.

Managing a Failing Noncritical Node

Use the following procedure to replace a failing node in either a CDH or a NoSQL cluster.

To manage a failing noncritical node:

1. Log in as root to the node where Mammoth is installed (typically node01).
2. Decommission the failing node. Replace *node_name* with the name of the failing node, such as bda1node07.

```
bdacli admin_cluster decommission node_name
```

3. Repair or replace the failed server.
4. From the Mammoth node as root, recommission the repaired or replaced server. Use the same name as the decommissioned node for *node_name*, such as bda1node07:

```
bdacli admin_cluster recommission node_name
```

See Also: *Oracle Big Data Appliance Owner's Guide* for the complete bdacli syntax

Stopping and Starting Oracle Big Data Appliance

This section describes how to shut down Oracle Big Data Appliance gracefully and restart it.

- [Prerequisites](#)
- [Stopping Oracle Big Data Appliance](#)
- [Starting Oracle Big Data Appliance](#)

Prerequisites

You must have `root` access. Passwordless SSH must be set up on the cluster, so that you can use the `dcli` utility.

To ensure that passwordless-ssh is set up:

1. Log in to the first node of the cluster as `root`.
2. Use a `dcli` command to verify it is working. This command should return the IP address and host name of every node in the cluster:

```
# dcli -C hostname
192.0.2.1: bda1node01.example.com
192.0.2.2: bda1node02.example.com
.
.
.
```

3. If you do not get these results, then set up `dcli` on the cluster:

```
# setup-root-ssh -C
```

See Also: *Oracle Big Data Appliance Owner's Guide* for details about these commands.

Stopping Oracle Big Data Appliance

Follow these procedures to shut down all Oracle Big Data Appliance software and hardware components.

Note: The following services stop automatically when the system shuts down. You do not need to take any action:

- Oracle Enterprise Manager agent
 - Auto Service Request agents
-
-

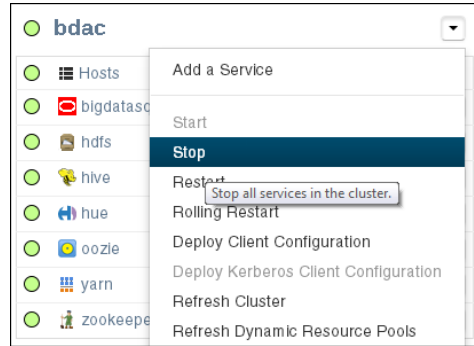
Task 1 Stopping All Managed Services

Use Cloudera Manager to stop the services it manages, including `flume`, `hbase`, `hdfs`, `hive`, `hue`, `mapreduce`, `oozie`, and `zookeeper`.

1. Log in to Cloudera Manager as the `admin` user.
See "[Managing Operations Using Cloudera Manager](#)" on page 2-3.
2. In the Status pane of the opening page, expand the menu for the cluster and click **Stop**, and then click **Stop** again when prompted to confirm. See [Figure 2-9](#).
To navigate to this page, click the **Home** tab, and then the **Status** subtab.
3. On the Command Details page, click **Close** when all processes are stopped.

4. In the same pane under Cloudera Management Services, expand the menu for the mgmt service and click **Stop**.
5. Log out of Cloudera Manager.

Figure 2–9 Stopping HDFS Services



Task 2 Stopping Cloudera Manager Server

Follow this procedure to stop Cloudera Manager Server.

1. Log in as root to the node where Cloudera Manager runs (initially node03).

Note: The remaining tasks presume that you are logged in to a server as root. You can enter the commands from any server by using the `dcli` command. This example runs the `pwd` command on node03 from any node in the cluster:

```
# dcli -c node03 pwd
```

2. Stop the Cloudera Manager server:

```
# service cloudera-scm-server stop
Stopping cloudera-scm-server: [ OK ]
```

3. Verify that the server is stopped:

```
# service cloudera-scm-server status
cloudera-scm-server is stopped
```

After stopping Cloudera Manager, you cannot access it using the web console.

Task 3 Stopping Oracle Data Integrator Agent

If Oracle Data Integrator is used on the cluster:

1. Check the status of the Oracle Data Integrator agent:

```
# dcli -C service odi-agent status
```

2. Stop the Oracle Data Integrator agent, if it is running:

```
# dcli -C service odi-agent stop
```

3. Ensure that the Oracle Data Integrator agent stopped running:

```
# dcli -C service odi-agent status
```

Task 4 Dismounting NFS Directories

All nodes share an NFS directory on node03, and additional directories may also exist. If a server with the NFS directory (/opt/exportdir) is unavailable, then the other servers hang when attempting to shut down. Thus, you must dismount the NFS directories first.

1. Locate any mounted NFS directories:

```
# dcli -C mount | grep sharedir
192.0.2.1: bdalnode03.example.com:/opt/exportdir on /opt/sharedir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.2: bdalnode03.example.com:/opt/exportdir on /opt/sharedir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.3: /opt/exportdir on /opt/sharedir type none (rw,bind)
.
.
.
```

The sample output shows a shared directory on node03 (192.0.2.3).

2. Dismount the shared directory:

```
# dcli -C umount /opt/sharedir
```

3. Dismount any custom NFS directories.

Task 5 Stopping the Servers

The Linux `shutdown -h` command powers down individual servers. You can use the `dcli -g` command to stop multiple servers.

1. Create a file that lists the names or IP addresses of the other servers in the cluster, that is, not including the one you are logged in to.
2. Stop the other servers:

```
# dcli -g filename shutdown -h now
```

For *filename*, enter the name of the file that you created in step 1.

3. Stop the server you are logged in to:

```
# shutdown -h now
```

Task 6 Stopping the InfiniBand and Cisco Switches

To stop the network switches, turn off a PDU or a breaker in the data center. The switches only turn off when power is removed.

The network switches do not have power buttons. They shut down only when power is removed.

To stop the switches, turn off all breakers in the two PDUs.

Starting Oracle Big Data Appliance

Follow these procedures to power up the hardware and start all services on Oracle Big Data Appliance.

Task 1 Powering Up Oracle Big Data Appliance

1. Switch on all 12 breakers on both PDUs.

2. Allow 4 to 5 minutes for Oracle ILOM and the Linux operating system to start on the servers.
3. If password-based, on-disk encryption is enabled, then log in and mount the Hadoop directories on those servers:

```
$ mount-hadoop-dirs
```

```
Enter password to mount Hadoop directories: password
```

If the servers do not start automatically, then you can start them locally by pressing the power button on the front of the servers, or remotely by using Oracle ILOM. Oracle ILOM has several interfaces, including a command-line interface (CLI) and a web console. Use whichever interface you prefer.

For example, you can log in to the web interface as `root` and start the server from the Remote Power Control page. The URL for Oracle ILOM is the same as for the host, except that it typically has a `-c` or `-ilom` extension. This URL connects to Oracle ILOM for `bda1node4`:

```
http://bda1node04-ilom.example.com
```

Task 2 Starting the HDFS Software Services

Use Cloudera Manager to start all the HDFS services that it controls.

1. Log in as `root` to the node where Cloudera Manager runs (initially `node03`).

Note: The remaining tasks presume that you are logged in to a server as `root`. You can enter the commands from any server by using the `dcli` command. This example runs the `pwd` command on `node03` from any node in the cluster:

```
# dcli -c node03 pwd
```

2. Verify that the Cloudera Manager started automatically on `node03`:

```
# service cloudera-scm-server status
cloudera-scm-server (pid 11399) is running...
```

3. If it is not running, then start it:

```
# service cloudera-scm-server start
```

4. Log in to Cloudera Manager as the `admin` user.

See ["Managing Operations Using Cloudera Manager"](#) on page 2-3.

5. In the Status pane of the opening page, expand the menu for the cluster and click **Start**, and then click **Start** again when prompted to confirm. See [Figure 2-9](#).

To navigate to this page, click the **Home** tab, and then the **Status** subtab.

6. On the Command Details page, click **Close** when all processes are started.
7. In the same pane under Cloudera Management Services, expand the menu for the `mgmt` service and click **Start**.
8. Log out of Cloudera Manager (optional).

Task 3 Starting Oracle Data Integrator Agent

If Oracle Data Integrator is used on this cluster:

1. Check the status of the agent:

```
# /opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/startcmd.sh  
OdiPingAgent [-AGENT_NAME=agent_name]
```

2. Start the agent:

```
# /opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/agent.sh  
[-NAME=agent_name] [-PORT=port_number]
```

Managing Oracle Big Data SQL

Oracle Big Data SQL is registered with Cloudera Manager as an add-on service. You can use Cloudera Manager to start, stop, and restart the Oracle Big Data SQL service or individual role instances, the same way as a CDH service.

Cloudera Manager also monitors the health of the Oracle Big Data SQL service, reports service outages, and sends alerts if the service is not healthy.

Adding and Removing the Oracle Big Data SQL Service

Oracle Big Data SQL is an optional service on Oracle Big Data Appliance. It may be installed with the other client software during the initial software installation or an upgrade. Use Cloudera Manager to determine whether it is installed. A separate license is required; Oracle Big Data SQL is not included with the Oracle Big Data Appliance license.

You cannot use Cloudera Manager to add or remove the Oracle Big Data SQL service from a CDH cluster on Oracle Big Data Appliance. Instead, log in to the server where Mammoth is installed (usually the first node of the cluster) and use the following commands in the `bdaccli` utility:

- To enable Oracle Big Data SQL

```
bdaccli enable big_data_sql
```

- To disable Oracle Big Data SQL:

```
bdaccli disable big_data_sql
```

See Also: *Oracle Big Data Appliance Owner's Guide*

Allocating Resources to Oracle Big Data SQL

You can modify the property values in a Linux kernel Control Group (Cgroup) to reserve resources for Oracle Big Data SQL.

To modify the resource management configuration settings:

1. Log in as admin to Cloudera Manager.
2. On the Home page, click **bigdatasql** from the list of services.
3. On the bigdatasql page, click **Configuration**.
4. Under Category, expand BDS Server Default Group and click **Resource Management**.
5. Modify the values of the following properties as required:
 - Cgroup CPU Shares
 - Cgroup I/O Weight
 - Cgroup Memory Soft Limit

- Cgroup Memory Hard Limit

See the Description column on the page for guidelines.

6. Click **Save Changes**.

7. From the Actions menu, click **Restart**.

Figure 2–10 shows the bigdatasql service configuration page.

Figure 2–10 *Modifying the Cgroup Settings for Oracle Big Data SQL*

The screenshot shows the 'bigdatasql' configuration interface. The 'Configuration' tab is selected. On the left, a sidebar lists categories: Service-Wide, BDS Server Default Group, and Resource Management (which is highlighted). Under 'Resource Management', there are sub-items: Advanced, Monitoring, Performance, and Resource Management. The main area displays a table of configuration properties:

Category	Property	Value
Service-Wide	Cgroup CPU Shares cpu.shares	500 Reset to the default value: 1024
BDS Server Default Group	Cgroup I/O Weight blkio.weight	250 Reset to the default value: 500
Resource Management	Cgroup Memory Soft Limit memory.soft_limit_in_bytes	-1 MiB Reset to the default value: -1 MiB
Resource Management	Cgroup Memory Hard Limit memory.limit_in_bytes	-1 MiB Reset to the default value: -1 MiB

See Also: ["Allocating Resources Among Services"](#) on page 2-10.

Security on Oracle Big Data Appliance

You can take precautions to prevent unauthorized use of the software and data on Oracle Big Data Appliance.

This section contains these topics:

- [About Predefined Users and Groups](#)
- [About User Authentication](#)
- [About Fine-Grained Authorization](#)
- [About On-Disk Encryption](#)
- [Port Numbers Used on Oracle Big Data Appliance](#)
- [About Puppet Security](#)

About Predefined Users and Groups

Every open-source package installed on Oracle Big Data Appliance creates one or more users and groups. Most of these users do not have login privileges, shells, or home directories. They are used by daemons and are not intended as an interface for individual users. For example, Hadoop operates as the `hdfs` user, MapReduce operates as `mapred`, and Hive operates as `hive`.

You can use the `oracle` identity to run Hadoop and Hive jobs immediately after the Oracle Big Data Appliance software is installed. This user account has login privileges, a shell, and a home directory.

Oracle NoSQL Database and Oracle Data Integrator run as the `oracle` user. Its primary group is `oinstall`.

Note: Do not delete, re-create, or modify the users that are created during installation, because they are required for the software to operate.

Table 2–9 identifies the operating system users and groups that are created automatically during installation of Oracle Big Data Appliance software for use by CDH components and other software packages.

Table 2–9 Operating System Users and Groups

User Name	Group	Used By	Login Rights
flume	flume	Apache Flume parent and nodes	No
hbase	hbase	Apache HBase processes	No
hdfs	hadoop	NameNode, DataNode	No
hive	hive	Hive metastore and server processes	No
hue	hue	Hue processes	No
mapred	hadoop	ResourceManager, NodeManager, Hive Thrift daemon	Yes
mysql	mysql	MySQL server	Yes
oozie	oozie	Oozie server	No
oracle	dba, oinstall	Oracle NoSQL Database, Oracle Loader for Hadoop, Oracle Data Integrator, and the Oracle DBA	Yes
puppet	puppet	Puppet parent (puppet nodes run as root)	No
sqoop	sqoop	Apache Sqoop metastore	No
svctag		Auto Service Request	No
zookeeper	zookeeper	ZooKeeper processes	No

About User Authentication

Oracle Big Data Appliance supports Kerberos security as a software installation option. See [Chapter 3](#) for details about setting up clients and users to access a Kerberos-protected cluster.

About Fine-Grained Authorization

The typical authorization model on Hadoop is at the HDFS file level, such that users either have access to all of the data in the file or none. In contrast, Apache Sentry integrates with the Hive and Impala SQL-query engines to provide fine-grained authorization to data and metadata stored in Hadoop.

Oracle Big Data Appliance automatically configures Sentry during software installation, beginning with Mammoth utility version 2.5.

See Also:

- Cloudera Manager Help
- *Managing Clusters with Cloudera Manager* at <https://www.cloudera.com/content/cloudera-content/cloudera-docs/CM4Ent/latest/Cloudera-Manager-Managing-Clusters/Managing-Clusters-with-Cloudera-Manager.html>

About On-Disk Encryption

On-disk encryption protects data that is at rest on disk. When on-disk encryption is enabled, Oracle Big Data Appliance automatically encrypts and decrypts data stored on disk. On-disk encryption does not affect user access to Hadoop data, although it can have a minor impact on performance.

Password-based encryption encodes Hadoop data based on a password, which is the same for all servers in a cluster. You can change the password at any time by using the `mammoth-reconfig update` command. See *Oracle Big Data Appliance Owner's Guide*.

If a disk is removed from a server, then the encrypted data remains protected until you install the disk in a server (the same server or a different one), startup the server, and provide the password. If a server is powered off and removed from an Oracle Big Data Appliance rack, then the encrypted data remains protected until you restart server and provide the password. You must enter the password after every startup of every server to enable access to the data. See "[Starting Oracle Big Data Appliance](#)" on page 2-22.

On-disk encryption is an option that you can select during the initial installation of the software by the Mammoth utility. You can also enable or disable on-disk encryption at any time by using either the `mammoth-reconfig` or `bdaccli` utilities.

See Also: *Oracle Big Data Appliance Owner's Guide*

Port Numbers Used on Oracle Big Data Appliance

[Table 2-10](#) identifies the port numbers that might be used in addition to those used by CDH.

To view the ports used on a particular server:

1. In Cloudera Manager, click the **Hosts** tab at the top of the page to display the Hosts page.
2. In the Name column, click a server link to see its detail page.
3. Scroll down to the Ports section.

See Also: For the full list of CDH port numbers, go to the Cloudera website at

http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM5/latest/Cloudera-Manager-Installation-Guide/cm5ig_config_ports.html

Table 2–10 Oracle Big Data Appliance Port Numbers

Service	Port
Automated Service Monitor (ASM)	30920
HBase master service (node01)	60010
MySQL Database	3306
Oracle Data Integrator Agent	20910
Oracle NoSQL Database administration	5001
Oracle NoSQL Database processes	5010 to 5020
Oracle NoSQL Database registration	5000
Port map	111
Puppet master service	8140
Puppet node service	8139
rpc.statd	668
ssh	22
xinetd (service tag)	6481

About Puppet Security

The puppet node service (puppetd) runs continuously as root on all servers. It listens on port 8139 for "kick" requests, which trigger it to request updates from the puppet master. It does not receive updates on this port.

The puppet master service (puppetmasterd) runs continuously as the puppet user on the first server of the primary Oracle Big Data Appliance rack. It listens on port 8140 for requests to push updates to puppet nodes.

The puppet nodes generate and send certificates to the puppet master to register initially during installation of the software. For updates to the software, the puppet master signals ("kicks") the puppet nodes, which then request all configuration changes from the puppet master node that they are registered with.

The puppet master sends updates only to puppet nodes that have known, valid certificates. Puppet nodes only accept updates from the puppet master host name they initially registered with. Because Oracle Big Data Appliance uses an internal network for communication within the rack, the puppet master host name resolves using /etc/hosts to an internal, private IP address.

Auditing Oracle Big Data Appliance

You can use Oracle Audit Vault and Database Firewall to create and monitor the audit trails for HDFS and MapReduce on Oracle Big Data Appliance.

This section describes the Oracle Big Data Appliance plug-in:

- [About Oracle Audit Vault and Database Firewall](#)
- [Setting Up the Oracle Big Data Appliance Plug-in](#)
- [Monitoring Oracle Big Data Appliance](#)

About Oracle Audit Vault and Database Firewall

Oracle Audit Vault and Database Firewall secures databases and other critical components of IT infrastructure in these key ways:

- Provides an integrated auditing platform for your enterprise.
- Captures activity on Oracle Database, Oracle Big Data Appliance, operating systems, directories, file systems, and so forth.
- Makes the auditing information available in a single reporting framework so that you can understand the activities across the enterprise. You do not need to monitor each system individually; you can view your computer infrastructure as a whole.

Audit Vault Server provides a web-based, graphic user interface for both administrators and auditors.

You can configure CDH/Hadoop clusters on Oracle Big Data Appliance as secured targets. The Audit Vault plug-in on Oracle Big Data Appliance collects audit and logging data from these services:

- **HDFS:** Who makes changes to the file system.
- **Hive DDL:** Who makes Hive database changes.
- **MapReduce:** Who runs MapReduce jobs that correspond to file access.
- **Oozie workflows:** Who runs workflow activities.

The Audit Vault plug-in is an installation option. The Mammoth utility automatically configures monitoring on Oracle Big Data Appliance as part of the software installation process.

See Also: For more information about Oracle Audit Vault and Database Firewall:

<http://www.oracle.com/technetwork/database/database-technologies/audit-vault-and-database-firewall/overview/index.html>

Setting Up the Oracle Big Data Appliance Plug-in

The Mammoth utility on Oracle Big Data Appliance performs all the steps needed to setup the plug-in, using information that you provide.

To set up the Audit Vault plug-in for Oracle Big Data Appliance:

1. Ensure that Oracle Audit Vault and Database Firewall Server Release 12.1.1 is up and running on the same network as Oracle Big Data Appliance.

See Also: *Oracle Audit Vault and Database Firewall Installation Guide*

2. Complete the Audit Vault Plug-in section of Oracle Big Data Appliance Configuration Generation Utility.
3. Install the Oracle Big Data Appliance software using the Mammoth utility. An Oracle representative typically performs this step.

You can also add the plug-in at a later time using either `bdaccli` or `mammoth-reconfig`. See *Oracle Big Data Appliance Owner's Guide*.

When the software installation is complete, the Audit Vault plug-in is installed on Oracle Big Data Appliance, and Oracle Audit Vault and Database Firewall is collecting its audit information. You do not need to perform any other installation steps.

See Also: *Oracle Big Data Appliance Owner's Guide* for using Oracle Big Data Appliance Configuration Generation Utility

Monitoring Oracle Big Data Appliance

After installing the plug-in, you can monitor Oracle Big Data Appliance the same as any other secured target. Audit Vault Server collects activity reports automatically.

The following procedure describes one type of monitoring activity.

To view an Oracle Big Data Appliance activity report:

1. Log in to Audit Vault Server as an auditor.
2. Click the **Reports** tab.
3. Under Built-in Reports, click **Audit Reports**.
4. To browse all activities, in the Activity Reports list, click the **Browse report data** icon for All Activity.
5. Add or remove the filters to list the events.

Event names include ACCESS, CREATE, DELETE, and OPEN.

6. Click the **Single row view** icon in the first column to see a detailed report.

[Figure 2–11](#) shows the beginning of an activity report, which records access to a Hadoop sequence file.

Figure 2–11 Activity Report in Audit Vault Server

All Activity Report	
Report View	< Row 0 of <input type="checkbox"/> Exclude Null Values <input type="checkbox"/> Displayed Columns
<div>Secured Target</div> <div> <div>Secured Target Name</div> <div>Secured Target Type</div> <div>Service Name</div> <div>Policy Name</div> <div>Class</div> <div>Other</div> </div>	
<div>Event</div> <div> <div>Server Time</div> <div>Event Time</div> <div>User Name</div> <div>Event Status</div> <div>Error Code</div> <div>Error Message</div> <div>Event Name</div> <div>Command Class</div> <div>Action Taken</div> <div>Threat Severity</div> <div>Log Cause</div> <div>Location</div> <div>Audit File</div> </div>	
<div>Target</div> <div> <div>Target Type</div> <div>Target Object</div> <div>Target Owner</div> </div>	

See Also: *Oracle Audit Vault and Database Firewall Auditor's Guide*

Collecting Diagnostic Information for Oracle Customer Support

If you need help from Oracle Support to troubleshoot CDH issues, then you should first collect diagnostic information using the `bdadiag` utility with the `cm` option.

To collect diagnostic information:

1. Log in to an Oracle Big Data Appliance server as `root`.
2. Run `bdadiag` with at least the `cm` option. You can include additional options on the command line as appropriate. See the *Oracle Big Data Appliance Owner's Guide* for a complete description of the `bdadiag` syntax.

```
# bdadiag cm
```

The command output identifies the name and the location of the diagnostic file.

3. Go to My Oracle Support at <http://support.oracle.com>.
4. Open a Service Request (SR) if you have not already done so.
5. Upload the `bz2` file into the SR. If the file is too large, then upload it to `sftp.oracle.com`, as described in the next procedure.

To upload the diagnostics to `ftp.oracle.com`:

1. Open an SFTP client and connect to `sftp.oracle.com`. Specify port 2021 and remote directory `/support/incoming/target`, where *target* is the folder name given to you by Oracle Support.
2. Log in with your Oracle Single Sign-on account and password.
3. Upload the diagnostic file to the new directory.
4. Update the SR with the full path and the file name.

See Also: My Oracle Support Note 549180.1 at
<http://support.oracle.com>

Supporting User Access to Oracle Big Data Appliance

This chapter describes how you can support users who run MapReduce jobs on Oracle Big Data Appliance or use Oracle Big Data Connectors. It contains these sections:

- [About Accessing a Kerberos-Secured Cluster](#)
- [Providing Remote Client Access to CDH](#)
- [Providing Remote Client Access to Hive](#)
- [Managing User Accounts](#)
- [Recovering Deleted Files](#)

About Accessing a Kerberos-Secured Cluster

Apache Hadoop is not an inherently secure system. It is protected only by network security. After a connection is established, a client has full access to the system.

To counterbalance this open environment, Oracle Big Data Appliance supports Kerberos security as a software installation option. Kerberos is a network authentication protocol that helps prevent malicious impersonation.

CDH provides these securities when configured to use Kerberos:

- The CDH master nodes, NameNodes, and JournalNodes resolve the group name so that users cannot manipulate their group memberships.
- Map tasks run under the identity of the user who submitted the job.
- Authorization mechanisms in HDFS and MapReduce help control user access to data.

If the Oracle Big Data Appliance cluster is secured with Kerberos, then you must take additional steps to authenticate a CDH client and individual users, as described in this chapter. Users must know their Kerberos user name, password, and realm.

[Table 3–1](#) describes some frequently used Kerberos commands. For more information, see the MIT Kerberos documentation.

Table 3–1 Kerberos User Commands

Command	Description
<code>kinit userid@realm</code>	Obtains a Kerberos ticket.
<code>klist</code>	Lists a Kerberos ticket if you have one already.
<code>kdestroy</code>	Invalidates a ticket before it expires.

Table 3–1 (Cont.) Kerberos User Commands

Command	Description
<code>kpasswd userid@realm</code>	Changes your password.

See Also:

- MIT Kerberos Documentation at
<http://web.mit.edu/kerberos/krb5-latest/doc/>
- *CDH4 Security Guide* at
<http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Security-Guide/CDH5-Security-Guide.html>

Providing Remote Client Access to CDH

Oracle Big Data Appliance supports full local access to all commands and utilities in Cloudera's Distribution including Apache Hadoop (CDH).

You can use a browser on any computer that has access to the client network of Oracle Big Data Appliance to access Cloudera Manager, Hadoop Map/Reduce Administration, the Hadoop Task Tracker interface, and other browser-based Hadoop tools.

To issue Hadoop commands remotely, however, you must connect from a system configured as a CDH client with access to the Oracle Big Data Appliance client network. This section explains how to set up a computer so that you can access HDFS and submit MapReduce jobs on Oracle Big Data Appliance.

See Also: My Oracle Support ID 1506203.1

Prerequisites

Ensure that you have met the following prerequisites:

- You must have these access privileges:
 - Sudo access to the client system
 - Login access to Cloudera Manager

If you do not have these privileges, then contact your system administrator for help.
- The client system must run an operating system that Cloudera supports for CDH5. See the *Cloudera CDH5 Installation Guide* at

http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Requirements-and-Supported-Versions/cdhrsv_os.html

- The client system must run Oracle JDK 1.7.0_25 or later.

To verify the version, use this command:

```
$ java -version
java version "1.7.0_65"
Java(TM) SE Runtime Environment (build 1.7.0_65-b17)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```


Installing a CDH Client on Any Supported Operating System

To install a CDH client on any operating system identified as supported by Cloudera, follow these instructions.

To install the CDH client software:

1. Log in to the client system.
2. If an earlier version of Hadoop is already installed, then remove it.
See the Cloudera documentation for removing an earlier CDH version at http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Installation-Guide/cdh5ig_to_cdh5_upgrade.html?scroll=topic_6_3_4_unique_1
3. Copy the CDH software from any node in a CDH cluster on Oracle Big Data Appliance. For example, the following file contains the CDH 5.3.0 software:
`/opt/oss/src/CDH/5.3.0-ol5/hadoop-2.5.0-cdh5.3.0.tar.gz`
4. Decompress the file into a permanent location, which will be the Hadoop home directory. The following command unzips the files into `hadoop-2.5.0-cdh5.3.0` in the current directory:
`tar -xvzf hadoop-2.5.0-cdh5.3.0.tar.gz`
5. Configure the CDH client. See "[Configuring a CDH Client for an Unsecured Cluster](#)" on page 3-3.

Configuring a CDH Client for an Unsecured Cluster

After installing CDH, you must configure it for use with Oracle Big Data Appliance.

The commands in this procedure that reference `HADOOP_HOME` are used to support older Hadoop clients that require this environment variable. The cluster uses YARN (MRv2) and does not use `HADOOP_HOME`. If no older clients access the cluster, then you can omit these commands.

To configure the Hadoop client:

1. Log in to the client system and download the MapReduce client configuration from Cloudera Manager. In this example, Cloudera Manager listens on port 7180 (the default) of `bda01node03.example.com`, and the configuration is stored in a file named `yarn-conf.zip`.

```
$ wget -O yarn-conf.zip
http://bda01node03.example.com:7180/cm/services/3/client-config
```
2. Unzip `mapreduce-config.zip` into a permanent location on the client system.

```
$ unzip yarn-config.zip
Archive:  yarn-config.zip
  inflating: yarn-conf/hadoop-env.sh
  inflating: yarn-conf/hdfs-site.xml
  inflating: yarn-conf/core-site.xml
  inflating: yarn-conf/mapred-site.xml
  inflating: yarn-conf/log4j.properties
  inflating: yarn-conf/yarn-site.xml
```

All files are stored in a subdirectory named `yarn-config`.

3. Set the symbolic links:

```
ln -s $HADOOP_HOME/../../../../bin-mapreduce1 $HADOOP_HOME/bin
ln -s $HADOOP_HOME/../../../../etc/hadoop-mapreduce1 $HADOOP_HOME/conf
rm -f $HADOOP_HOME/lib/slf4j-log4j*jar
```

4. Open `hadoop-env.sh` in a text editor and set the environment variables to the actual paths on your system:

```
export HADOOP_HOME=hadoop-home-dir/share/hadoop/mapreduce1
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=yarn-conf-dir
export JAVA_HOME=/usr/java/version
alias hadoop=$HADOOP_HOME/bin/hadoop
alias hdfs=$HADOOP_HOME/../../../../bin/hdfs
```

5. Make a backup copy of the Hadoop configuration files:

```
# cp /full_path/yarn-conf /full_path/yarn-conf-bak
```

6. Overwrite the existing configuration files with the downloaded configuration files in Step 2.

```
# cd /full_path/yarn-conf
# cp * /usr/lib/hadoop/conf
```

Configuring a CDH Client for a Kerberos-Secured Cluster

Follow these steps to enable the CDH client to work with a secure CDH cluster.

To configure a CDH client for Kerberos:

1. Log in to the system where you created the CDH client.
2. Install the Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files:

- a. Download the files for your Java version:

Java 6:

<http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html>

Java 7:

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

- b. Decompress the downloaded file. This example unzips JCE-7:

```
$ unzip UnlimitedJCEPolicyJDK7.zip
Archive:  UnlimitedJCEPolicyJDK7.zip
  creating:  UnlimitedJCEPolicy/
  inflating:  UnlimitedJCEPolicy/US_export_policy.jar
  inflating:  UnlimitedJCEPolicy/local_policy.jar
  inflating:  UnlimitedJCEPolicy/README.txt
```

Note: The JCE-6 files unzip into a directory named `jce` instead of `UnlimitedJCEPolicy`.

- c. Copy the unzipped files into the Java security directory. For example:

```
$ cp UnlimitedJCEPolicy/* /usr/java/latest/jre/lib/security/
```

3. Follow the steps for configuring an unsecured client.

See ["Configuring a CDH Client for an Unsecured Cluster"](#) on page 3-3.

4. Ensure that you have a user ID on the CDH cluster that had been added to the Kerberos realm.

See ["Creating Hadoop Cluster Users"](#) on page 3-8.

5. On the CDH client system, create a file named `krb5.conf` in the `$HADOOP_CONF_DIR` directory. Enter configuration settings like the following, using values appropriate for your installation for the server names, domain, and realm:

```
[libdefaults]
    default_realm = EXAMPLE.COM
    dns_lookup_realm = false
    dns_lookup_kdc = false
    clocks skew = 3600
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true

[realms]
    EXAMPLE.COM = {
        kdc = bda01node01.example:88
        admin_server = bda01node07:749
        default_domain = example.com
    }

[domain_realm]
    .com = EXAMPLE.COM
```

6. Activate the new configuration file:

```
export KRB5_CONFIG=$HADOOP_CONF_DIR/krb5.conf
export HADOOP_OPTS="-Djava.security.krb5.conf=$HADOOP_CONF_DIR/krb5.conf"
export KRB5CCNAME=$HADOOP_CONF_DIR/krb5cc_$USER
```

7. Verify that you have access to the Oracle Big Data Appliance cluster.

See ["Verifying Access to a Cluster from the CDH Client"](#) on page 3-5.

Verifying Access to a Cluster from the CDH Client

Follow this procedure to ensure that you have access to the Oracle Big Data Appliance cluster.

To verify cluster access:

1. To access a Kerberos-protected CDH cluster, first obtain a ticket granting ticket (TGT):

```
$ kinit userid@realm
```

2. Verify that you can access HDFS on Oracle Big Data Appliance from the client, by entering a simple Hadoop file system command like the following:

```
$ hadoop fs -ls /user
Found 6 items
```

```
drwxr-xr-x - jdoe      hadoop          0 2014-04-03 00:08 /user/jdoe
drwxrwxrwx - mapred    hadoop          0 2014-04-02 23:25 /user/history
drwxr-xr-x - hive      supergroup     0 2014-04-02 23:27 /user/hive
```

```
drwxrwxr-x - impala impala 0 2014-04-03 10:45 /user/impala
drwxr-xr-x - oozie hadoop 0 2014-04-02 23:27 /user/oozie
drwxr-xr-x - oracle hadoop 0 2014-04-03 11:49 /user/oracle
```

Check the output for HDFS users defined on Oracle Big Data Appliance, and not on the client system. You should see the same results as you would after entering the command directly on Oracle Big Data Appliance.

- 3. Submit a MapReduce job. You must be logged in to the client system under the same user name as your HDFS user name on Oracle Big Data Appliance.

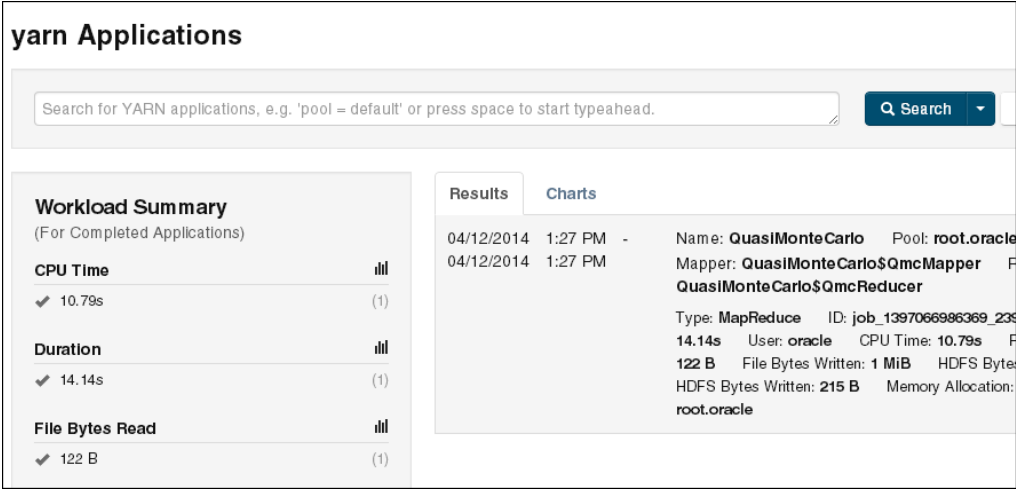
The following example calculates the value of *pi*:

```
$ hadoop jar
/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2.5.0-cdh5.2.1.jar pi 10
1000000
Number of Maps = 10
Samples per Map = 1000000
Wrote input for Map #0
Wrote input for Map #1
.
.
.
Job Finished in 12.403 seconds
Estimated value of Pi is 3.14158440000000000000
```

- 4. Use Cloudera Manager to verify that the job ran on Oracle Big Data Appliance instead of the local system. Select **mapreduce Jobs** from the Activities menu for a list of jobs.

Figure 3–1 shows the job created by the previous example.

Figure 3–1 Monitoring a YARN Job in Cloudera Manager



Providing Remote Client Access to Hive

Follow this procedure to provide remote client access to Hive.

To set up a Hive client:

- 1. Set up a CDH client. See "Providing Remote Client Access to CDH" on page 3-2.

2. Log in to the client system and download the Hive client configuration from Cloudera Manager. In this example, Cloudera Manager listens on port 7180 (the default) of `bda01node03.example.com`, and the configuration is stored in a file named `hive-conf.zip`.

```
$ wget -O hive-conf.zip
http://bda01node03.example.com:7180/cmfservices/5/client-config
Length: 1283 (1.3K) [application/zip]
Saving to: 'hive-conf.zip'

100%[=====] 1,283      --.-K/s   in 0.001s

2014-05-15 08:19:06 (2.17 MB/s) - 'hive-conf.zip' saved [1283/1283]
```

3. Unzip the file into a permanent installation directory, which will be the Hive configuration directory:

```
$ unzip hive-conf.zip
Archive:  hive-conf.zip
  inflating: hive-conf/hive-env.sh
  inflating: hive-conf/hive-site.xml
```

4. Download the Hive software from the Cloudera website:

```
$ wget http://archive.cloudera.com/cdh5/cdh/5/hive-0.12.0-cdh5.0.0.tar.gz
Length: 49637596 (47M) [application/x-gzip]
Saving to: 'hive-0.12.0-cdh5.0.0.tar.gz'

100%[=====] 49,637,596   839K/s   in 47s

2014-05-15 08:22:18 (1.02 MB/s) - 'hive-0.12.0-cdh5.0.0.tar.gz' saved
[49637596/49637596]
```

5. Decompress the file into a permanent installation directory, which will be the Hive home directory. The following command unzips the files into the current directory in a subdirectory named `hive-0.12.0-cdh5.0.0`:

```
$ tar -xvzf hive-0.12.0-cdh5.0.0.tar.gz
hive-0.12.0-cdh5.0.0/
hive-0.12.0-cdh5.0.0/examples/
.
.
.
```

6. Set the following variables, replacing *hive-home-dir* and *hive-conf-dir* with the directories you created in steps 3 and 5.

```
export HIVE_HOME=hive-home-dir
export HIVE_CONF_DIR=hive-conf-dir
alias hive=$HIVE_HOME/bin/hive
```

The following steps test whether you successfully set up a Hive client.

To verify Hive access:

1. To access a Kerberos-protected CDH cluster, first obtain a ticket granting ticket (TGT):

```
$ kinit userid@realm
```

2. Open the Hive console:

```
$ hive
Logging initialized using configuration in
jar:file:/usr/lib/hive/lib/hive-common-0.12.0-cdh5.0.0.jar!/hive-log4j.properties
```

```
es
Hive history file=/tmp/oracle/hive_job_log_
e10527ee-9637-4c08-9559-a2e5cea6cef1_831268640.txt
hive>
```

3. List all tables:

```
hive> show tables;
OK
src
```

Managing User Accounts

This section describes how to create users who can access HDFS, MapReduce, and Hive. It contains the following topics:

- [Creating Hadoop Cluster Users](#)
- [Providing User Login Privileges \(Optional\)](#)

Creating Hadoop Cluster Users

When creating user accounts, define them as follows:

- To run MapReduce jobs, users must either be in the `hadoop` group or be granted the equivalent permissions.
- To create and modify tables in Hive, users must either be in the `hive` group or be granted the equivalent permissions.
- To create Hue users, open Hue in a browser and click the User Admin icon. See ["Using Cloudera Hue to Interact With Hadoop"](#) on page 2-6.

Creating Users on an Unsecured Cluster

To create a user on an unsecured Hadoop cluster:

1. Open an ssh connection as the `root` user to a noncritical node (node04 to node18).
2. Create the user's home directory:

```
# sudo -u hdfs hadoop fs -mkdir /user/user_name
```

You use `sudo` because the HDFS super user is `hdfs` (not `root`).

3. Change the ownership of the directory:

```
# sudo -u hdfs hadoop fs -chown user_name:hadoop /user/user_name
```

4. Verify that the directory is set up correctly:

```
# hadoop fs -ls /user
```

5. Create the operating system user across all nodes in the cluster:

```
# dcli useradd -G hadoop,hive[,group_name...] -m user_name
```

In this syntax, replace *group_name* with an existing group and *user_name* with the new name.

6. Verify that the operating system user belongs to the correct groups:

```
# dcli id user_name
```

7. Verify that the user's home directory was created on all nodes:

```
# dcli ls /home | grep user_name
```

[Example 3-1](#) creates a user named `jdoe` with a primary group of `hadoop` and an additional group of `hive`.

Example 3-1 Creating a Hadoop User

```
# sudo -u hdfs hadoop fs -mkdir /user/jdoe
# sudo -u hdfs hadoop fs -chown jdoe:hadoop /user/jdoe
# hadoop fs -ls /user
Found 5 items
drwx----- - hdfs      supergroup          0 2013-01-16 13:50 /user/hdfs
drwxr-xr-x - hive      supergroup          0 2013-01-16 12:58 /user/hive
drwxr-xr-x - jdoe      jdoe                0 2013-01-18 14:04 /user/jdoe
drwxr-xr-x - oozie     hadoop              0 2013-01-16 13:01 /user/oozie
drwxr-xr-x - oracle    hadoop              0 2013-01-16 13:01 /user/oracle
# dcli useradd -G hadoop,hive -m jdoe
# dcli id jdoe
bda1node01: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),127(hive),123(hadoop)
bda1node02: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
bda1node03: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
.
.
.
# dcli ls /home | grep jdoe
bda1node01: jdoe
bda1node02: jdoe
bda1node03: jdoe
```

Creating Users on a Secured Cluster

To create a user on a Kerberos-secured cluster:

1. Connect to Kerberos as the HDFS principal and execute the following commands, replacing `jdoe` with the actual user name:

```
hdfs dfs -mkdir /user/jdoe
hdfs dfs -chown jdoe /user/jdoe
dcli -C useradd -G hadoop,hive -m jdoe
hash=$(echo "hadoop" | openssl passwd -1 -stdin)
dcli -C "usermod --pass='$hash' jdoe"
```

2. Log in to the key distribution center (KDC) and add a principal for the user. In the following example, replace `jdoe`, `bda01node01`, and `example.com` with the correct user name, server name, domain, and realm.

```
ssh -l root bda01node01.example.com kadmin.local
add_principal user_name@EXAMPLE.COM
```

Providing User Login Privileges (Optional)

Users do not need login privileges on Oracle Big Data Appliance to run MapReduce jobs from a remote client. However, for those who want to log in to Oracle Big Data Appliance, you must set a password. You can set or reset a password the same way.

To set a user password across all Oracle Big Data Appliance servers:

1. Create a Hadoop cluster user as described in ["Creating Hadoop Cluster Users"](#) on page 3-8..

2. Confirm that the user does not have a password:

```
# dcli passwd -S user_name
bda1node01.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
bda1node02.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
bda1node03.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
```

If the output shows either "Empty password" or "Password locked," then you must set a password.

3. Set the password:

```
hash=$(echo 'password' | openssl passwd -1 -stdin); dcli "usermod
--pass='$hash' user_name"
```

4. Confirm that the password is set across all servers:

```
# dcli passwd -S user_name
bda1node01.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bda1node02.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bda1node03.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
```

See Also:

- *Oracle Big Data Appliance Owner's Guide* for information about dcli.
- The Linux man page for the full syntax of the useradd command.

Recovering Deleted Files

CDH provides an optional trash facility, so that a deleted file or directory is moved to a trash directory for a set period, instead of being deleted immediately from the system. By default, the trash facility is enabled for HDFS and all HDFS clients.

Restoring Files from the Trash

When the trash facility is enabled, you can easily restore files that were previously deleted.

To restore a file from the trash directory:

1. Check that the deleted file is in the trash. The following example checks for files deleted by the oracle user:

```
$ hadoop fs -ls .Trash/Current/user/oracle
Found 1 items
-rw-r--r--  3 oracle hadoop  242510990 2012-08-31 11:20
/user/oracle/.Trash/Current/user/oracle/ontime_s.dat
```

2. Move or copy the file to its previous location. The following example moves ontime_s.dat from the trash to the HDFS /user/oracle directory.

```
$ hadoop fs -mv .Trash/Current/user/oracle/ontime_s.dat /user/oracle/ontime_
s.dat
```


Changing the Trash Interval

The **trash interval** is the minimum number of minutes that a file remains in the trash directory before being deleted permanently from the system. The default value is 1 day (24 hours).

To change the trash interval:

1. Open Cloudera Manager. See ["Managing Operations Using Cloudera Manager"](#) on page 2-3.
2. On the Home page under Status, click **hdfs**.
3. On the hdfs page, click the Configuration subtab, and then select **View and Edit**.
4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Default Group. See [Figure 3-2](#).
5. Click the current value, and enter a new value in the pop-up form.
6. Click **Save Changes**.
7. Expand the Actions menu at the top of the page and choose **Restart**.
8. Open a connection as root to a node in the cluster.
9. Deploy the new configuration:

```
dccli -C bdagetcclientconfig
```

[Figure 3-2](#) shows the Filesystem Trash Interval property in Cloudera Manager.

Figure 3-2 HDFS Property Settings in Cloudera Manager

The screenshot shows the Cloudera Manager interface for the HDFS configuration. At the top, there's a 'Good Health' status indicator and an 'Actions' menu. Below this is a navigation bar with tabs for Status, Instances, Commands, Configuration (selected), Audits, Charts Library, and File Browser. Under Configuration, there are sub-tabs for Cache Statistics, Replication, and Web UI. The main content area is titled 'Configuration' and features a search bar with 'trash' entered. Below the search bar, a green banner indicates '3 validation checks'. A table lists the configuration properties:

Category	Property	Value	Description
Gateway Default Group	Use Trash	<input checked="" type="checkbox"/> Reset to the default value: false HDFS Trash is enabled.	Move deleted files to the trash so that they can be recovered if necessary. This client side configuration takes effect only if the HDFS service-wide trash is disabled (NameNode Filesystem Trash Interval set to 0) and is ignored otherwise. The trash is not automatically emptied when enabled with this configuration.
NameNode Default Group	Filesystem Trash Interval	1 day(s) default value Trash checkpointing is on	Number of minutes between trash checkpoints. Also controls the number of minutes after which a trash checkpoint directory is deleted. To disable the trash feature, enter 0.

Disabling the Trash Facility

The trash facility on Oracle Big Data Appliance is enabled by default. You can change this configuration for a cluster. When the trash facility is disabled, deleted files and directories are not moved to the trash. They are not recoverable.

Completely Disabling the Trash Facility

The following procedure disables the trash facility for HDFS. When the trash facility is completely disabled, the client configuration is irrelevant.

To completely disable the trash facility:

1. Open Cloudera Manager. See ["Managing Operations Using Cloudera Manager"](#) on page 2-3.
2. On the Home page under Status, click **hdfs**.
3. On the hdfs page, click the Configuration subtab, and then select **View and Edit**.
4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Default Group. See [Figure 3-2](#).
5. Click the current value, and enter a value of 0 (zero) in the pop-up form.
6. Click **Save Changes**.
7. Expand the Actions menu at the top of the page and choose **Restart**.

Disabling the Trash Facility for Local HDFS Clients

All HDFS clients that are installed on Oracle Big Data Appliance are configured to use the trash facility. An **HDFS client** is any software that connects to HDFS to perform operations such as listing HDFS files, copying files to and from HDFS, and creating directories.

You can use Cloudera Manager to change the local client configuration setting, although the trash facility is still enabled.

Note: If you do not want any clients to use the trash, then you can completely disable the trash facility. See ["Completely Disabling the Trash Facility"](#) on page 3-12.

To disable the trash facility for local HDFS clients:

1. Open Cloudera Manager. See ["Managing Operations Using Cloudera Manager"](#) on page 2-3.
2. On the Home page under Status, click **hdfs**.
3. On the hdfs page, click the **Configuration** subtab, and then select **View and Edit**.
4. Search for or scroll down to the Filesystem Trash Interval property under Gateway Default Group. See [Figure 3-2](#).
5. Search for or scroll down to the Use Trash property under Client Settings. See [Figure 3-2](#).
6. Deselect the Use Trash check box.
7. Click **Save Changes**. This setting is used to configure all new HDFS clients downloaded to Oracle Big Data Appliance.
8. Open a connection as **root** to a node in the cluster.
9. Deploy the new configuration:

```
dccli -C bdagetcclientconfig
```

Disabling the Trash Facility for a Remote HDFS Client

Remote HDFS clients are typically configured by downloading and installing a CDH client, as described in ["Providing Remote Client Access to CDH"](#) on page 3-2. Oracle SQL Connector for HDFS and Oracle R Advanced Analytics for Hadoop are examples of remote clients.

To disable the trash facility for a remote HDFS client:

1. Open a connection to the system where the CDH client is installed.
2. Open `/etc/hadoop/conf/hdfs-site.xml` in a text editor.
3. Set the trash interval to zero:

```
<property>
  <name>fs.trash.interval</name>
  <value>0</value>
</property>
```

4. Save the file.

Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance

This chapter provides information about optimizing communications between Oracle Exadata Database Machine and Oracle Big Data Appliance. It describes how you can configure Oracle Exadata Database Machine to use InfiniBand alone, or SDP over InfiniBand, to communicate with Oracle Big Data Appliance.

This chapter contains the following sections:

- [About Optimizing Communications](#)
- [Prerequisites for Optimizing Communications](#)
- [Specifying the InfiniBand Connections to Oracle Big Data Appliance](#)
- [Specifying the InfiniBand Connections to Oracle Exadata Database Machine](#)
- [Enabling SDP on Exadata Database Nodes](#)
- [Creating an SDP Listener on the InfiniBand Network](#)

About Optimizing Communications

Oracle Exadata Database Machine and Oracle Big Data Appliance use Ethernet by default, although typically they are also connected by an InfiniBand network. Ethernet communications are much slower than InfiniBand. After you configure Oracle Exadata Database Machine to communicate using InfiniBand, it can obtain data from Oracle Big Data Appliance many times faster than before.

Moreover, client applications that run on Oracle Big Data Appliance and push the data to Oracle Database can use Sockets Direct Protocol (SDP) for an additional performance boost. SDP is a standard communication protocol for clustered server environments, providing an interface between the network interface card and the application. By using SDP, applications place most of the messaging burden upon the network interface card, which frees the CPU for other tasks. As a result, SDP decreases network latency and CPU utilization, and thereby improves performance.

About Applications that Pull Data Into Oracle Exadata Database Machine

Oracle SQL Connector for Hadoop Distributed File System (HDFS) is an example of an application that pulls data into Oracle Exadata Database Machine. The connector enables an Oracle external table to access data stored in either HDFS files or a Hive table.

The external table provide access to the HDFS data. You can use the external table for querying HDFS data or for loading it into an Oracle database table.

Oracle SQL Connector for HDFS functions as a Hadoop client running on the database servers in Oracle Exadata Database Machine.

If you use Oracle SQL Connector for HDFS or another tool that pulls the data into Oracle Exadata Database Machine, then for the best performance, you should configure the system to use InfiniBand. See ["Specifying the InfiniBand Connections to Oracle Big Data Appliance"](#) on page 4-2.

See Also : *Oracle Big Data Connectors User's Guide* for information about Oracle SQL Connector for HDFS

About Applications that Push Data Into Oracle Exadata Database Machine

Oracle Loader for Hadoop is an example of an application that pushes data into Oracle Exadata Database Machine. The connector is an efficient and high-performance loader for fast movement of data from a Hadoop cluster into a table in an Oracle database. You can use it to load data from Oracle Big Data Appliance to Oracle Exadata Database Machine.

Oracle Loader for Hadoop functions as a database client running on the Oracle Big Data Appliance. It must make database connections from Oracle Big Data Appliance to Oracle Exadata Database Machine over the InfiniBand network. Use of Sockets Direct Protocol (SDP) for these database connections further improves performance.

If you use Oracle Loader for Hadoop or another tool that pushes the data into Oracle Exadata Database Machine, then for the best performance, you should configure the system to use SDP over InfiniBand as described in this chapter.

See Also : *Oracle Big Data Connectors User's Guide* for information about Oracle Loader for Hadoop

Prerequisites for Optimizing Communications

Oracle Big Data Appliance and Oracle Exadata Database Machine racks must be cabled together using InfiniBand cables. The IP addresses must be unique across all racks and use the same subnet for the InfiniBand network.

See Also:

- *Oracle Big Data Appliance Owner's Guide* about multirack cabling
- *Oracle Big Data Appliance Owner's Guide* about IP addresses and subnets

Specifying the InfiniBand Connections to Oracle Big Data Appliance

You can configure Oracle Exadata Database Machine to use the InfiniBand IP addresses of the Oracle Big Data Appliance servers. Otherwise, the default network is Ethernet. Use of the InfiniBand network improves the performance of all data transfers between Oracle Big Data Appliance and Oracle Exadata Database Machine.

To identify the Oracle Big Data Appliance InfiniBand IP addresses:

1. If you have not done so already, install a CDH client on Oracle Exadata Database Machine. See ["Providing Remote Client Access to CDH"](#) on page 3-2.
2. Obtain a list of private host names and InfiniBand IP addresses for all Oracle Big Data Appliance servers.

An Oracle Big Data Appliance rack can have 6, 12, or 18 servers.

3. Log in to Oracle Exadata Database Machine with `root` privileges.
4. Edit `/etc/hosts` on Oracle Exadata Database Machine and add the Oracle Big Data Appliance host names and InfiniBand IP addresses. The following example shows the sequential IP numbering:

```
192.168.8.1      bda1node01.example.com  bda1node01
192.168.8.2      bda1node02.example.com  bda1node02
192.168.8.3      bda1node03.example.com  bda1node03
192.168.8.4      bda1node04.example.com  bda1node04
192.168.8.5      bda1node05.example.com  bda1node05
192.168.8.6      bda1node06.example.com  bda1node06
```

5. Check `/etc/nsswitch.conf` for a line like the following:

```
hosts:      files dns
```

Ensure that the line does not reverse the order (`dns files`); if it does, your additions to `/etc/hosts` will not be used. Edit the file if necessary.

6. Ping all Oracle Big Data Appliance servers. Ensure that ping completes and shows the InfiniBand IP addresses.

```
# ping bda1node01.example.com
PING bda1node01.example.com (192.168.8.1) 56(84) bytes of data.
64 bytes from bda1node01.example.com (192.168.8.1): icmp_seq=1 ttl=50 time=20.2
ms
.
.
.
```

7. Run CDH locally on Oracle Exadata Database Machine and test HDFS functionality by uploading a large file to an Oracle Big Data Appliance server. Check that your network monitoring tools (such as `sar`) show I/O activity on the InfiniBand devices.

To upload a file, use syntax like the following, which copies `localfile.dat` to the HDFS `testdir` directory on node05 of Oracle Big Data Appliance:

```
hadoop fs -put localfile.dat hdfs://bda1node05.example.com/testdir/
```

Specifying the InfiniBand Connections to Oracle Exadata Database Machine

You can configure Oracle Big Data Appliance to use the InfiniBand IP addresses of the Oracle Exadata Database Machine servers. This configuration supports applications on Oracle Big Data Appliance that must connect to Oracle Exadata Database Machine.

To identify the Oracle Exadata Database Machine InfiniBand IP addresses:

1. Obtain a list of private host names and InfiniBand IP addresses for all Oracle Exadata Database Machine servers.
2. Log in to Oracle Big Data Appliance with `root` privileges.
3. Edit `/etc/hosts` on Oracle Big Data Appliance and add the Oracle Exadata Database Machine host names and InfiniBand IP addresses.
4. Check `/etc/nsswitch.conf` for a line like the following:

```
hosts:      files dns
```

Ensure that the line does not reverse the order (dns files); if it does, your additions to /etc/hosts will not be used. Edit the file if necessary.

5. Restart the dnsmasq service:

```
# service dnsmasq restart
```

6. Ping all Oracle Exadata Database Machine servers. Ensure that ping completes and shows the InfiniBand IP addresses.
7. Test the connection by downloading a large file to an Oracle Exadata Database Machine server. Check that your network monitoring tools (such as sar) show I/O activity on the InfiniBand devices.

To download a file, use syntax like the following, which copies a file named mydata.json to the dm01cel08 storage server:

```
$ scp mydata.json oracle@dm01cel08-priv.example.com:mybigdata.json
oracle@dm01cel08-priv.example.com's password: password
```

Enabling SDP on Exadata Database Nodes

SDP improves the performance of client applications that run on Oracle Big Data Appliance and push large data loads to Oracle Database on Oracle Exadata Database Machine.

The following procedure describes how to enable SDP on the database nodes in an Oracle Exadata Database Machine running Oracle Linux. You must also configure your application on a job-by-job basis to use SDP.

To enable SDP on Oracle Exadata Database Machine:

1. Open /etc/infiniband/openib.conf file in a text editor, and add the following line:

```
set: SDP_LOAD=yes
```

2. Save these changes and close the file.
3. To enable both SDP and TCP, open /etc/ofed/libsdp.conf in a text editor, and add the use both rule:

```
use both server * :
use both client * :
```

4. Save these changes and close the file.
5. Open /etc/modprobe.conf file in a text editor, and add this setting:

```
options ib_sdp sdp_zcopy_thresh=0 rcv_poll=0
```
6. Save these changes and close the file.
7. Replicate these changes across all database nodes in the Oracle Exadata Database Machine rack.
8. Restart all database nodes for the changes to take effect.
9. If you have multiple Oracle Exadata Database Machine racks, then repeat these steps on all of them.

To specify SDP protocol for a load job:

1. Add JVM options to the HADOOP_OPTS environment variable to enable JDBC SDP export:


```
HADOOP_OPTS="-Doracle.net.SDP=true -Djava.net.preferIPv4Stack=true"
```

2. In either the Hadoop command or the configuration file for the job, set the `mapred.child.java.opts` configuration property to enable the child task JVMs for SDP.

For example, use these options in the command line for a MapReduce job:

```
-D mapred.child.java.opts="-Doracle.net.SDP=true  
-Djava.net.preferIPv4Stack=true"
```

3. Configure standard Ethernet communications for the job.

For example, Oracle Loader for Hadoop reads the value of the `oracle.hadoop.loader.connection.url` property from a job configuration file. The value has this syntax:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=  
  (ADDRESS=(PROTOCOL=TCP) (HOST=hostName) (PORT=portNumber)))  
  (CONNECT_DATA=(SERVICE_NAME=serviceName)))
```

Replace *hostName*, *portNumber*, and *serviceName* with the appropriate values to identify the SDP listener on your Oracle Exadata Database Machine.

4. Configure the Oracle listener on Exadata to support the SDP protocol and bind it to a specific port address (such as 1522).

For example, Oracle Loader for Hadoop reads the value of the `oracle.hadoop.loader.connection.oci_url` property from a job configuration file. The value has this syntax:

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=SDP)  
  (HOST=hostName) (PORT=portNumber))  
  (CONNECT_DATA=(SERVICE_NAME=serviceName)))
```

Creating an SDP Listener on the InfiniBand Network

To add a listener for the Oracle Big Data Appliance connections coming in on the InfiniBand network, first add a network resource for the InfiniBand network with virtual IP addresses.

Note: This example lists two nodes for an Oracle Exadata Database Machine quarter rack. If you have an Oracle Exadata Database Machine half or full rack, you must repeat node-specific lines for each node in the cluster.

1. Edit `/etc/hosts` on each node in the Exadata rack to add the virtual IP addresses for the InfiniBand network. Make sure that these IP addresses are not in use. For example:

```
# Added for Listener over IB  
192.168.10.21 dm01db01-ibvip.example.com dm01db01-ibvip  
192.168.10.22 dm01db02-ibvip.example.com dm01db02-ibvip
```

2. As the root user, create a network resource on one database node for the InfiniBand network. For example:

```
# /u01/app/grid/product/12.1.0.1/bin/srvctl add network -k 2 -S  
192.168.10.0/255.255.255.0/bondib0
```

3. Verify that the network was added correctly with a command like the following examples:

```
# /u01/app/grid/product/12.1.0.1/bin/crsctl stat res -t | grep net
ora.net1.network
ora.net2.network -- Output indicating new Network resource
```

or

```
# /u01/app/grid/product/12.1.0.1/bin/srvctl config network -k 2
Network exists: 2/192.168.10.0/255.255.255.0/bondib0, type static -- Output
indicating Network resource on the 192.168.10.0 subnet
```

4. Add the virtual IP addresses on the network created in Step 2, for each node in the cluster. For example:

```
# srvctl add vip -n dm01db01 -A dm01db01-ibvip/255.255.255.0/bondib0 -k 2
#
# srvctl add vip -n dm01db02 -A dm01db02-ibvip/255.255.255.0/bondib0 -k 2
```

5. As the oracle user who owns Grid Infrastructure Home, add a listener for the virtual IP addresses created in Step 4.

```
# srvctl add listener -l LISTENER_IB -k 2 -p TCP:1522,/SDP:1522
```

6. For each database that will accept connections from the middle tier, modify the listener_networks init parameter to allow load balancing and failover across multiple networks (Ethernet and InfiniBand). You can either enter the full TNSNAMES syntax in the initialization parameter or create entries in tnsnames.ora in the \$ORACLE_HOME/network/admin directory. The TNSNAMES.ORA entries must exist in GRID_HOME. The following example first updates tnsnames.ora.

Complete this step on each node in the cluster with the correct IP addresses for that node. LISTENER_IBREMOTE should list all other nodes that are in the cluster. DBM_IB should list all nodes in the cluster.

Note: The database instance reads the TNSNAMES only on startup. Thus, if you modify an entry that is referred to by any init.ora parameter (LISTENER_NETWORKS), then you must either restart the instance or issue an ALTER SYSTEM SET LISTENER_NETWORKS command for the modifications to take affect by the instance.

```
DBM =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP) (HOST = dm01-scan) (PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = dbm)
  )
)

DBM_IB =
(DESCRIPTION =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP) (HOST = dm01db01-ibvip) (PORT = 1522))
  (ADDRESS = (PROTOCOL = TCP) (HOST = dm01db02-ibvip) (PORT = 1522))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = dbm)
  )
)
```

```

LISTENER_IBREMOTE =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST = dm01db02-ibvip.mycompany.com) (PORT = 1522))
))

LISTENER_IBLOCAL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST = dm01db01-ibvip.mycompany.com) (PORT = 1522))
(ADDRESS = (PROTOCOL = SDP) (HOST = dm01db01-ibvip.mycompany.com) (PORT = 1523))
))

LISTENER_IPLOCAL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST = dm0101-vip.mycompany.com) (PORT = 1521))
))

LISTENER_IPREMOTE =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST = dm01-scan.mycompany.com) (PORT = 1521))
))

```

7. Connect to the database instance as sysdba.
8. Modify the `listener_networks` init parameter by using the SQL `ALTER SYSTEM` command:

```

SQL> alter system set listener_networks=
      '((NAME=network2) (LOCAL_LISTENER=LISTENER_IBLOCAL)
        (REMOTE_LISTENER=LISTENER_IBREMOTE))',
      '((NAME=network1) (LOCAL_LISTENER=LISTENER_IPLOCAL)
        (REMOTE_LISTENER=LISTENER_IPREMOTE))' scope=both;

```

9. On the Linux command line, use the `srvctl` command to restart `LISTENER_IB` to implement the modification in Step 7:

```

# srvctl stop listener -l LISTENER_IB
# srvctl start listener -l LISTENER_IB

```


Part II

Oracle Big Data Appliance Software

This part describes the software that is available only on Oracle Big Data Appliance. It contains the following chapters:

- [Chapter 5, "Optimizing MapReduce Jobs Using Perfect Balance"](#)

Optimizing MapReduce Jobs Using Perfect Balance

This chapter describes how you can shorten the run time of some MapReduce jobs by using Perfect Balance. It contains the following sections:

- [What is Perfect Balance?](#)
- [Application Requirements](#)
- [Getting Started with Perfect Balance](#)
- [Analyzing a Job's Reducer Load](#)
- [About Configuring Perfect Balance](#)
- [Running a Balanced MapReduce Job Using Perfect Balance](#)
- [About Perfect Balance Reports](#)
- [About Chopping](#)
- [Troubleshooting Jobs Running with Perfect Balance](#)
- [Using the Perfect Balance API](#)
- [About the Perfect Balance Examples](#)
- [Perfect Balance Configuration Property Reference](#)

What is Perfect Balance?

The Perfect Balance feature of Oracle Big Data Appliance distributes the reducer load in a MapReduce application so that each reduce task does approximately the same amount of work. While the default Hadoop method of distributing the reduce load is appropriate for many jobs, it does not distribute the load evenly for jobs with significant data skew.

Data skew is an imbalance in the load assigned to different reduce tasks. The **load** is a function of:

- The number of keys assigned to a reducer.
- The number of records and the number of bytes in the values per key.

The total run time for a job is extended, to varying degrees, by the time that the reducer with the greatest load takes to finish. In jobs with a skewed load, some reducers complete the job quickly, while others take much longer. Perfect Balance can significantly shorten the total run time by distributing the load evenly, enabling all reducers to finish at about the same time.

Your MapReduce job can be written using either the `mapred` or `mapreduce` APIs; Perfect Balance supports both of them.

About Balancing Jobs Across Map and Reduce Tasks

A typical Hadoop job has map and reduce tasks. Hadoop distributes the *mapper* workload uniformly across Hadoop Distributed File System (HDFS) and across map tasks, while preserving the data locality. In this way, it reduces skew in the mappers.

Hadoop also hashes the map-output keys uniformly across all *reducers*. This strategy works well when there are many more keys than reducers, and each key represents a very small portion of the workload. However, it is not effective when the mapper output is concentrated into a small number of keys. Hashing these keys results in skew and does not work in applications like sorting, which require range partitioning.

Perfect Balance distributes the load evenly across reducers by first sampling the data, optionally chopping large keys into two or more smaller keys, and using a load-aware partitioning strategy to assign keys to reduce tasks.

Ways to Use Perfect Balance Features

You can choose from two methods of running Perfect Balance features:

- **Perfect Balance:** You can run a job without changing your application code by properly configuring Perfect Balance. This is the preferred method and appropriate for most jobs.

This method is described in ["Running a Balanced MapReduce Job Using Perfect Balance"](#) on page 5-10, and is the primary focus of this chapter.
- **Perfect Balance API:** You can add the code to run Perfect Balance to your application code. Use this method when your application setup code must run before using Perfect Balance. Otherwise, you should use the previous method, which requires no change to your code.

This method is described in ["Using the Perfect Balance API"](#) on page 5-15.

Perfect Balance Components

Perfect Balance has these components:

- **Job Analyzer:** Gathers and reports statistics about the MapReduce job so that you can determine whether to use Perfect Balance.
- **Counting Reducer:** Provides additional statistics to the Job Analyzer to help gauge the effectiveness of Perfect Balance.
- **Load Balancer:** Runs before the MapReduce job to generate a static partition plan, and reconfigures the job to use the plan. The balancer includes a user-configurable, progressive sampler that stops sampling the data as soon as it can generate a good partitioning plan.

Application Requirements

To use Perfect Balance successfully, your application must meet the following requirements:

- The job is distributive, so that splitting a group of records associated with a reduce key does not produce incorrect results for the application.

To balance a load, Perfect Balance subpartitions the values of large reduce keys and sends each subpartition to a different reducer. This distribution contrasts with the standard Hadoop practice of sending all values for a single reduce key to the same reducer. Your application must be able to handle output from the reducers that is not fully aggregated, so that it does not produce incorrect results.

This partitioning of values is called **chopping**. Applications that support chopping have **distributive reduce functions**. See ["About Chopping"](#) on page 5-13.

If your application is not distributive, then you can still run Perfect Balance after disabling the key-splitting feature. The job still benefits from using Perfect Balance, but the load is not as evenly balanced as it is when key splitting is in effect. See the [oracle.hadoop.balancer.keyLoad.minChopBytes](#) configuration property to disable key splitting.

- This release does not support combiners. Perfect Balance detects the presence of combiners and does not balance when they are present.

Getting Started with Perfect Balance

Take the following steps to use Perfect Balance:

1. Ensure that your application meets the requirements listed in ["Application Requirements"](#) on page 5-2.
2. Log in to the server where you will submit the job.
3. Run the examples provided with Perfect Balance to become familiar with the product. All examples shown in this chapter are based on the shipped examples and use the same data set. See ["About the Perfect Balance Examples"](#) on page 5-17.
4. Set the following variables using the Bash export command:
 - `BALANCER_HOME`: Set to the Perfect Balance installation directory, such as `/opt/oracle/orabalancer-2.2.0-h2` on Oracle Big Data Appliance (optional). The examples in this chapter use this variable, and you can also define it for your convenience. Perfect Balance does not require `BALANCER_HOME`.
 - `HADOOP_CLASSPATH`: Add `${BALANCER_HOME}/jlib/orabalancer-2.2.0.jar` and `${BALANCER_HOME}/jlib/commons-math-2.2.jar` to the existing value. Also add the JAR files for your application.

To enable Perfect Balance, add `${BALANCER_HOME}/jlib/orabalancerclient-2.2.0.jar`. (API mode does not use this JAR file.)
 - `HADOOP_USER_CLASSPATH_FIRST`: Set to `true` to enable Perfect Balance; API mode does not need this variable.

You do not need to set these variables unless you are using Perfect Balance.

5. Run Job Analyzer without the balancer and use the generated report to decide whether the job is a good candidate for using Perfect Balance.
See ["Analyzing a Job's Reducer Load"](#) on page 5-4.
6. Decide which configuration properties to set. Create a configuration file or enter the settings individually in the `hadoop` command.
See ["About Configuring Perfect Balance"](#) on page 5-9.
7. Run the job using Perfect Balance.
See ["Running a Balanced MapReduce Job Using Perfect Balance"](#) on page 5-10.

8. Use the Job Analyzer report to evaluate the effectiveness of using Perfect Balance. See ["Reading the Job Analyzer Report"](#) on page 5-8.
9. Modify the job configuration properties as desired before rerunning the job with Perfect Balance. See ["About Configuring Perfect Balance"](#) on page 5-9.

Analyzing a Job's Reducer Load

Job Analyzer is a component of Perfect Balance that identifies imbalances in a load, and how effective Perfect Balance is in correcting the imbalance when actually running the job. This section contains the following topics:

- [About Job Analyzer](#)
- [Running Job Analyzer as a Standalone Utility](#)
- [Running Job Analyzer Using Perfect Balance](#)
- [Reading the Job Analyzer Report](#)

About Job Analyzer

You can use Job Analyzer to decide whether a job is a candidate for load balancing with Perfect Balance. Job Analyzer uses the output logs of a MapReduce job to generate a simple report with statistics like the elapsed time and the load for each reduce task. By default, it uses the standard Hadoop counters displayed by the JobTracker user interface, but organizes the data to emphasize the relative performance and load of the reduce tasks, so that you can more easily interpret the results.

If the report shows that the data is skewed (that is, the reducers processed very different loads and the run times varied widely), then the application is a good candidate for Perfect Balance.

Methods of Running Job Analyzer

You can choose between two methods of running Job Analyzer:

- **As a standalone utility:** Job Analyzer runs against existing job output logs. This is a good choice when you want to analyze a job that previously ran.
- **While using Perfect Balance:** Job Analyzer runs against the output logs for the current job running with Perfect Balance. This is a good choice when you want to analyze the current job.

Running Job Analyzer as a Standalone Utility

As a standalone utility, Job Analyzer provides a quick way to analyze the reduce load of a previously run job.

To run Job Analyzer as a standalone utility:

1. Log in to the server where you will run Job Analyzer.
2. Locate the output logs from the job to analyze:
 - **YARN clusters:** Set `oracle.hadoop.balancer.application_id` to the job ID of the job you want to analyze. YARN is the default on Oracle Big Data Appliance.

You can obtain the job ID from the YARN Resource Manager web interface. Click the application ID of a job, and then click **Tracking URL**. The job ID typically begins with "job_".

Alternately, if you already ran Perfect Balance or Job Analyzer on this job, you can read the job ID from the `application_id` file generated by Perfect Balance in its report directory (`outdir/_balancer` by default).

- **MRv1 clusters:** Set `mapred.output.dir` to the output directory of the job you want to analyze.
3. Run the Job Analyzer utility as described in ["Job Analyzer Utility Syntax"](#) on page 5-5.
 4. View the Job Analyzer report in a browser.

Job Analyzer Utility Example

[Example 5-1](#) runs a script that sets the required variables, uses the MapReduce job logs for a job with an application ID of `job_1396563311211_0947`, and creates the report in the default location. It then copies the HTML version of the report from HDFS to the `/home/jdoe` local directory and opens the report in a browser.

If you want to run this example in YARN, then replace the application ID with the application ID of the job. The application ID of the job looks like this example: `job_1396563311211_0947`.

If you use MRv1 instead of YARN, then set `mapred.output.dir` instead of `oracle.hadoop.balancer.application_id`.

Example 5-1 Running the Job Analyzer Utility

```
$ cat runja.sh

BALANCER_HOME=/opt/oracle/orabalancer-2.2.0-h2
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancer-2.2.0.jar:${BALANCER_HOME}/jlib/commons-math-2.2.jar:$HADOOP_CLASSPATH
export HADOOP_USER_CLASSPATH_FIRST=true

# Command on YARN cluster
hadoop jar orabalancer.jar oracle.hadoop.balancer.tools.JobAnalyzer \
-D oracle.hadoop.balancer.application_id=job_1396563311211_0947

$ sh ./runja.sh
$
$ hadoop fs -get jdoe_nobal_outdir/_balancer/jobanalyzer-report.html /home/jdoe
$ cd /home/jdoe
$ firefox jobanalyzer-report.html
```

Job Analyzer Utility Syntax

The following is the syntax to run the Job Analyzer utility:

For YARN:

```
hadoop jar ${BALANCER_HOME}/jlib/orabalancer-2.2.0.jar
oracle.hadoop.balancer.tools.JobAnalyzer \
-D oracle.hadoop.balancer.application_id=job_number \
[ja_report_path]
```

For MRv1:

```
hadoop jar ${BALANCER_HOME}/jlib/orabalancer-2.2.0.jar
```

```
oracle.hadoop.balancer.tools.JobAnalyzer \  
-D mapred.output.dir=job_output_dir \  
[ja_report_path]
```

job_number

The application ID previously assigned to the job. YARN only.

job_output_dir

An HDFS directory where the job files are stored from previously executing the application. MRv1 only.

ja_report_path

An HDFS directory where Job Analyzer creates its report (optional). The default directory is `job_output_dir/_balancer`.

Running Job Analyzer Using Perfect Balance

When you run a job using Perfect Balance, you can configure it to run Job Analyzer automatically. This section contains the following topics:

- [Running Job Analyzer Using Perfect Balance](#)
- [Collecting Additional Metrics](#)

Running Job Analyzer Using Perfect Balance

Follow these steps to run Job Analyzer using Perfect Balance:

1. Log in to the server where you will submit the job that uses Perfect Balance.
2. Set up Perfect Balance by taking the steps in "[Getting Started with Perfect Balance](#)" on page 5-3.
3. To enable Job Analyzer, set the `oracle.hadoop.balancer.autoAnalyze` configuration property to one of these values:
 - `BASIC_REPORT`: Enables Job Analyzer. If you set `oracle.hadoop.balancer.autoBalance` to `true`, then Perfect Balance automatically sets `oracle.hadoop.balancer.autoAnalyze` to `BASIC_REPORT`.
 - `REDUCER_REPORT`: Configures Job Analyzer to collect additional load statistics. See "[Collecting Additional Metrics](#)" on page 5-7.
4. Decide which additional configuration properties to set, if any.
See "[Perfect Balance Configuration Property Reference](#)" on page 5-18.
5. Run the job.

Example 5-2 runs a script that sets the required variables, uses Perfect Balance to run a job with Job Analyzer and without load balancing, and creates the report in the default location. It then copies the HTML version of the report from HDFS to the `/home/jdoe` local directory and opens the report in a browser. The output includes warnings, which you can ignore.

Example 5-2 Running Job Analyzer with Perfect Balance

```
$ cat ja_nobalance.sh  
  
# set up perfect balance  
BALANCER_HOME=/opt/oracle/orabalancer-2.2.0-h2  
export HADOOP_CLASSPATH=${BALANCER_  
HOME}/jlib/orabalancerclient-2.2.0.jar:${BALANCER_
```

```

HOME}/jlib/orabalancer-2.2.0.jar:${BALANCER_
HOME}/jlib/commons-math-2.2.jar:${HADOOP_CLASSPATH}
export HADOOP_USER_CLASSPATH_FIRST=true

# run the job
hadoop jar application_jarfile.jar ApplicationClass \
-D application_config_property \
-D mapreduce.input.fileinputformat.inputdir=jdoe_application/input \
-D mapreduce.output.fileoutputformat.outputdir=jdoe_nobal_outdir \
-D mapreduce.job.name=nobal \
-D mapreduce.job.reduces=10 \
-D oracle.hadoop.balancer.autoBalance=false \
-D oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT \
-conf application_config_file.xml

$ sh ja_nobalance.sh
14/04/14 14:52:42 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 14:52:42 INFO mapreduce.JobSubmitter: number of splits:5
14/04/14 14:52:42 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_
1397066986369_3478
14/04/14 14:52:43 INFO impl.YarnClientImpl: Submitted application application_
1397066986369_3478
.
.
.
File Input Format Counters
Bytes Read=112652976
File Output Format Counters
Bytes Written=384974202

$ hadoop fs -get jdoe_nobal_outdir/_balancer/jobanalyzer-report.html /home/jdoe
$ cd /home/jdoe
$ firefox jobanalyzer-report.html

```

Collecting Additional Metrics

The Job Analyzer report includes the load metrics for each key, if you set the `oracle.hadoop.balancer.autoAnalyze` property to `REDUCER_REPORT`. This additional information provides a more detailed picture of the load for each reducer, with metrics that are not available in the standard Hadoop counters.

The Job Analyzer report also compares its predicted load with the actual load. The difference between these values measures how effective Perfect Balance was in balancing the job.

Job Analyzer might recommend key load coefficients for the Perfect Balance key load model, based on its analysis of the job load. To use these recommended coefficients when running a job with Perfect Balance, set the `oracle.hadoop.balancer.linearKeyLoad.feedbackDir` property to the directory containing the Job Analyzer report of a previously analyzed run of the job.

If the report contains recommended coefficients, then Perfect Balance automatically uses them. If Job Analyzer encounters an error while collecting the additional metrics, then the report does not contain the additional metrics.

Use the `feedbackDir` property when you do not know the values of the load model coefficients for a job, but you have the Job Analyzer output from a previous run of the job. Then you can set the value of `feedbackDir` to the directory where that output is stored. The values recommended from those files typically perform better than the

Perfect Balance default values, because the recommended values are based on an analysis of your job's load.

Alternately, if you already know good values of the load model coefficients for your job, you can set the load model properties:

- `oracle.hadoop.balancer.linearKeyLoad.byteWeight`
- `oracle.hadoop.balancer.linearKeyLoad.keyWeight`
- `oracle.hadoop.balancer.linearKeyLoad.rowWeight`

Running the job with these coefficients results in a more balanced job.

Reading the Job Analyzer Report

Job Analyzer writes its report in two formats: HTML for you, and XML for Perfect Balance. You can open the report in a browser, either directly in HDFS or after copying it to the local file system

To open a Job Analyzer report in HDFS in a browser:

1. Open the HDFS web interface on port 50070 of a NameNode node (node01 or node02), using a URL like the following:

```
http://bdainode01.example.com:50070
```

2. From the Utilities menu, choose **Browse the File System**.
3. Navigate to the `job_output_dir/_balancer` directory.

To open a Job Analyzer report in the local file system in a browser:

1. Copy the report from HDFS to the local file system:

```
$ hadoop fs -get job_output_dir/_balancer/jobanalyzer-report.html /home/jdoe
```

2. Switch to the local directory:

```
$ cd /home/jdoe
```

3. Open the file in a browser:

```
$ firefox jobanalyzer-report.html
```

When inspecting the Job Analyzer report, look for indicators of skew such as:

- The execution time of some reducers is longer than others.
- Some reducers process more records or bytes than others.
- Some map output keys have more records than others.
- Some map output records have more bytes than others.

[Figure 5-1](#) shows the beginning of the analyzer report for the inverted index (`invindx`) example. It displays the key load coefficient recommendations, because this job ran with the appropriate configuration settings. See ["Collecting Additional Metrics"](#) on page 5-7.

The task IDs are links to tables that show the analysis of specific tasks, enabling you to drill down for more details from the first, summary table.

This example uses an extremely small data set, but notice the differences between tasks 7 and 8: The input records range from 3% to 29%, and their corresponding elapsed times range from 5 to 15 seconds. This variation indicates skew.

Figure 5–1 Job Analyzer Report for Unbalanced Inverted Index Job

Job Information		Time Information	
Job Name	invindx	Map Phase	00:00:29
Job Id	job_1405207264024_0122	Reduce Phase	00:00:19
Start Time	2014-07-22 15:03:39	Shuffle	00:00:04
Finish Time	2014-07-22 15:04:35	Merge	00:00:08
		Reduce	00:00:15
		Job	00:00:56

Reduce Tasks Metrics Summary

Task ID	Time			%Load
	Start	Finish	Elapsed	Observed
0	15:04:15	15:04:27	00:00:11	12
1	15:04:15	15:04:25	00:00:09	7
2	15:04:15	15:04:25	00:00:09	8
3	15:04:15	15:04:25	00:00:09	7
4	15:04:15	15:04:27	00:00:11	10
5	15:04:15	15:04:28	00:00:12	13
6	15:04:16	15:04:24	00:00:07	4
7	15:04:16	15:04:23	00:00:06	3
8	15:04:16	15:04:35	00:00:18	29
9	15:04:16	15:04:26	00:00:09	8

Reduce Tasks Metrics

Task ID	ElapsedTime			Input								Output			
	Shuffle	Merge	Reduce	Shuffle Bytes		Keys		Records		ValueBytes		Records		Bytes	
				count	%	count	%	count	%	count	%	count	%	count	%
0	00:00:03	00:00:02	00:00:05	20,987,702	12	13	13	2,300,118	12	43,702,242	12	1,698,171	12	48,022,226	12
1	00:00:03	00:00:01	00:00:04	15,630,571	9	12	12	1,438,876	7	27,338,644	7	1,309,838	9	36,424,492	9
2	00:00:03	00:00:02	00:00:04	14,682,583	8	7	7	1,503,328	8	28,563,232	8	1,220,301	9	33,206,894	9
3	00:00:03	00:00:01	00:00:04	14,977,081	9	10	10	1,450,603	7	27,561,457	7	1,253,630	9	33,638,218	9
4	00:00:03	00:00:02	00:00:05	21,175,220	12	11	11	2,073,885	10	39,403,815	10	1,768,698	13	51,191,915	13
5	00:00:03	00:00:03	00:00:05	19,664,004	11	10	10	2,580,271	13	49,025,149	13	1,497,856	11	40,709,092	11
6	00:00:03	00:00:01	00:00:03	8,867,408	5	9	9	775,690	4	14,738,110	4	740,944	5	20,503,323	5
7	00:00:03	00:00:00	00:00:02	5,912,478	3	6	6	531,247	3	10,093,693	3	494,798	4	13,336,988	3
8	00:00:03	00:00:06	00:00:08	35,890,632	21	11	11	5,824,950	29	110,674,050	29	2,527,641	18	69,087,902	18
9	00:00:03	00:00:02	00:00:04	16,252,928	9	11	11	1,521,032	8	28,899,608	8	1,359,917	10	38,889,633	10
Total	00:00:04	00:00:08	00:00:15	174,040,607	-	100	-	20,000,000	-	380,000,000	-	13,871,794	-	385,010,683	-

About Configuring Perfect Balance

Perfect Balance uses the standard Hadoop methods of specifying configuration properties in the command line. You can use the `-conf` option to identify a configuration file, or the `-D` option to specify individual properties. All Perfect Balance configuration properties have default values, and so setting them is optional.

"[Perfect Balance Configuration Property Reference](#)" on page 5-18 lists the configuration properties in alphabetical order with a full description. The following are functional groups of properties.

Perfect Balance Basic Properties

- `oracle.hadoop.balancer.autoAnalyze`
- `oracle.hadoop.balancer.autoBalance`

Job Analyzer Properties

- `oracle.hadoop.balancer.application_id`
- `oracle.hadoop.balancer.tools.jobConfPath`
- `oracle.hadoop.balancer.tools.jobHistoryPath`
- `oracle.hadoop.balancer.tools.writeKeyBytes`

Key Chopping Properties

- `oracle.hadoop.balancer.enableSorting`
- `oracle.hadoop.balancer.keyLoad.minChopBytes`

Load Balancing Properties

- `oracle.hadoop.balancer.confidence`
- `oracle.hadoop.balancer.maxLoadFactor`
- `oracle.hadoop.balancer.maxSamplesPct`
- `oracle.hadoop.balancer.minSplits`

Load Model Properties

- `oracle.hadoop.balancer.linearKeyLoad.feedbackDir`
- `oracle.hadoop.balancer.linearKeyLoad.byteWeight`
- `oracle.hadoop.balancer.linearKeyLoad.keyWeight`
- `oracle.hadoop.balancer.linearKeyLoad.rowWeight`

MapReduce-Related Properties

- `oracle.hadoop.balancer.useMapreduceApi`
- `oracle.hadoop.balancer.inputFormat.mapred.map.tasks`
- `oracle.hadoop.balancer.inputFormat.mapred.max.split.size`

Partition Report Properties

- `oracle.hadoop.balancer.report.override`
- `oracle.hadoop.balancer.reportPath`
- `oracle.hadoop.balancer.tmpDir`

Sampler Properties

- `oracle.hadoop.balancer.minSplits`
- `oracle.hadoop.balancer.numThreads`
- `oracle.hadoop.balancer.runMode`
- `oracle.hadoop.balancer.useClusterStats`

Running a Balanced MapReduce Job Using Perfect Balance

Perfect Balance does not require you to make any changes to your application code. It works by automatically running Perfect Balance for your job when you submit it to Hadoop for execution.

To run a job with Perfect Balance:

1. Log in to the server where you will submit the job.

2. Set up Perfect Balance by following the steps in ["Getting Started with Perfect Balance"](#) on page 5-3.
3. Configure the job with these Perfect Balance properties:
 - To enable balancing, set `oracle.hadoop.balancer.autoBalance` to true. This setting also runs Job Analyzer. Load balancing is not enabled by default.
 - To allow Job Analyzer to collect additional metrics, set `oracle.hadoop.balancer.autoAnalyze` to `REDUCER_REPORT`.
See ["Collecting Additional Metrics"](#) on page 5-7.
 - Decide which additional configuration properties to set, if any.
See ["About Configuring Perfect Balance"](#) on page 5-9.
4. Run your job as usual, using the following syntax:

```
bin/hadoop jar application_jarfile.jar ApplicationClass \
-D application_config_property \
-D oracle.hadoop.balancer.autoBalance=true \
-D other_perfect_balance_config_property \
-conf application_config_file.xml \
-conf perfect_balance_config_file.xml
```

You do not need to make any code changes to your application. You can provide Perfect Balance configuration properties either on the command line or in a configuration file. You can also combine Perfect Balance properties and MapReduce properties in the same configuration file. ["About Configuring Perfect Balance"](#) on page 5-9.

[Example 5-3](#) runs a script named `pb_balance.sh`, which sets up Perfect Balance for a job, and then runs the job. The key load metric properties are set to the values recommended in the Job Analyzer report shown in [Figure 5-1](#).

Example 5-3 Running a Job Using Perfect Balance

```
$ cat pb_balance.sh
```

```
#setup perfect balance as described in Getting Started with Perfect Balance
BALANCER_HOME=/opt/oracle/orabalancer-2.2.0-h2
export HADOOP_CLASSPATH=${BALANCER_
HOME}/jlib/orabalancerclient-2.2.0.jar:${BALANCER_
HOME}/jlib/orabalancer.jar:${BALANCER_HOME}/jlib/commons-math-2.2.jar:$HADOOP_
CLASSPATH
export HADOOP_USER_CLASSPATH_FIRST=true

# setup optional properties like java heap size and garbage collector
export HADOOP_CLIENT_OPTS="-Xmx1024M ${HADOOP_CLIENT_OPTS}"

# run the job with balancing and job analyzer enabled
hadoop jar application_jarfile.jar ApplicationClass
-D application_config_property \
-D mapreduce.input.fileinputformat.inputdir=jdoe_application/input \
-D mapreduce.output.fileoutputformat.outputdir=jdoe_outdir \
-D mapreduce.job.name="autoinvoke" \
-D mapreduce.job.reduces=10 \
-D oracle.hadoop.balancer.autoBalance=true \
-D oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT \
-D oracle.hadoop.balancer.linearKeyLoad.keyWeight=93.98 \
-D oracle.hadoop.balancer.linearKeyLoad.rowWeight=0.001126 \
-D oracle.hadoop.balancer.linearKeyLoad.byteWeight=0.0 \
```

```
-conf application_config_file.xml
```

```
$ sh ./pb_balance.sh
14/04/14 14:59:42 INFO balancer.Balancer: Creating balancer
14/04/14 14:59:42 INFO balancer.Balancer: Starting Balancer
14/04/14 14:59:43 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 14:59:46 INFO balancer.Balancer: Balancer completed
14/04/14 14:59:47 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 14:59:47 INFO mapreduce.JobSubmitter: number of splits:5
14/04/14 14:59:47 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_
1397066986369_3500
14/04/14 14:59:47 INFO impl.YarnClientImpl: Submitted application application_
1397066986369_3500
14/04/14 14:59:47 INFO mapreduce.Job: The url to track the job:
.
.
.
Map-Reduce Framework
Map input records=1000000
Map output records=20000000
Map output bytes=872652976
Map output materialized bytes=175650573
Input split bytes=580
Combine input records=0
Combine output records=0
Reduce input groups=106
Reduce shuffle bytes=175650573
Reduce input records=20000000
Reduce output records=13871794
Spilled Records=60000000
Shuffled Maps =50
Failed Shuffles=0
Merged Map outputs=50
GC time elapsed (ms)=1573
CPU time spent (ms)=242850
Physical memory (bytes) snapshot=6789033984
Virtual memory (bytes) snapshot=24548044800
Total committed heap usage (bytes)=11921457152
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=112652976
File Output Format Counters
Bytes Written=384974202
```

About Perfect Balance Reports

Perfect Balance generates these reports when it runs a job:

- **Job Analyzer report:** Contains various indicators about the distribution of the load in a job. The report is saved in HTML for you, and XML for Perfect Balance to use. The report is always named `jobanalyzer-report.html` and `-.xml`. See ["Reading the Job Analyzer Report"](#) on page 5-8.

- **Partition report:** Identifies the keys that are assigned to the various mappers. This report is saved in XML for Perfect Balance to use; it does not contain information of use to you. The report is named `${job_output_dir}/_balancer/orabalancer_report.xml`. It is only generated for balanced jobs.
- **Reduce key metric reports:** Perfect Balance generates a report for each file partition, when the appropriate configuration properties are set. The reports are saved in XML for Perfect Balance to use; they do not contain information of use to you. They are named `${job_output_dir}/_balancer/ReduceKeyMetricList-attempt_jobid_taskid_task_attemptid.xml`. They are generated only when the counting reducer is used (that is, `oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT` when using Perfect Balance, or a call to the `Balancer.configureCountingReducer` method when using the API.

See ["Collecting Additional Metrics"](#) on page 5-7.

The reports are stored by default in the job output directory (`${mapreduce.output.fileoutputformat.outputdir}` in YARN and `${mapred.output.dir}` in MRv1). Following is the structure of that directory:

```
job_output_directory
/_SUCCESS
/_balancer
  ReduceKeyMetricList-attempt_201305031125_0016_r_000000_0.xml
  ReduceKeyMetricList-attempt_201305031125_0016_r_000001_0.xml
  .
  .
  .
  jobanalyzer-report.html
  jobanalyzer-report.xml
  orabalancer_report.xml
/part-r-00000
/part-r-00001
.
.
.
```

About Chopping

To balance a load, Perfect Balance might subpartition the values of a single reduce key and send each subpartition to a different reducer. This partitioning of values is called **chopping**.

Selecting a Chopping Method

You can configure how Perfect Balance chops the values by setting the `oracle.hadoop.balancer.enableSorting` configuration property:

- **Chopping by hash partitioning:** Set `enableSorting=false` when sorting is not required. This is the default chopping strategy.
- **Chopping by sorting:** Set `enableSorting=true` to sort the values in each subpartition and order them across all subpartitions. In any parallel sort job, each task sort the rows within the task. The job must ensure that the values in reduce task 2 are greater than values in reduce task 1, the values in reduce task 3 are greater than the values in reduce task 2, and so on. The job generates multiple files containing data in sorted order, instead of one large file with sorted data.

For example, if a key is chopped into three subpartitions, and the subpartitions are sent to reducers 5, 8 and 9, then the values for that key in reducer 9 are greater than all values for that key in reducer 8, and the values for that key in reducer 8 are greater than all values for that key in reducer 5. When `enableSorting=true`, Perfect Balance ensures this ordering across reduce tasks.

If an application requires that the data is aggregated across files, then you can disable chopping by setting `oracle.hadoop.balancer.keyLoad.minChopBytes=-1`. Perfect Balance still offers performance gains by combining smaller reduce keys, called **bin packing**.

How Chopping Impacts Applications

If a MapReduce job aggregates the data by reduce key, then each reduce task aggregates the values for each key within that task. However, when chopping is enabled in Perfect Balance, the rows associated with a reduce key might be in different reduce tasks, leading to partial aggregation. Thus, values for a reduce key are aggregated within a reduce task, but not across reduce tasks. (The values for a reduce key across reduce tasks can be sorted, as discussed in ["Selecting a Chopping Method"](#) on page 5-13.)

When complete aggregation is required, you can disable chopping. Alternatively, you can examine the application that consumes the output of your MapReduce job. The application might work well with partial aggregation.

For example, a search engine might read in parallel the output from a MapReduce job that creates an inverted index. The output of a reduce task is a list of words, and for each word, a list of documents in which the word occurs. The word is the key, and the list of documents is the value. With partial aggregation, some words have multiple document lists instead of one aggregated list. Multiple lists are convenient for the search engine to consume in parallel. A parallel search engine might even require document lists to be split instead of aggregated into one list. See ["About the Perfect Balance Examples"](#) on page 5-17 for a Hadoop job that creates an inverted index from a document collection.

As another example, Oracle Loader for Hadoop loads data from multiple files to the correct partition of a target table. The load step is faster when there are multiple files for a reduce key, because they enable a higher degree of parallelism than loading from one file for a reduce key.

Troubleshooting Jobs Running with Perfect Balance

If you get Java "out of heap space" or "GC overhead limit exceeded" errors on the client node while running the Perfect Balance sampler, then increase the client JVM heap size for the job.

Use the Java JVM `-Xmx` option. You can specify client JVM options before running the Hadoop job, by setting the `HADOOP_CLIENT_OPTS` variable:

```
$ export HADOOP_CLIENT_OPTS="-Xmx1024M $HADOOP_CLIENT_OPTS"
```

Setting `HADOOP_CLIENT_OPTS` changes the JVM options only on the client node. It does not change JVM options in the map and reduce tasks. See the `invindx` script for an example of setting this variable.

Setting `HADOOP_CLIENT_OPTS` is sufficient to increase the heap size for the sampler, regardless of whether `oracle.hadoop.balancer.runMode` is set to local or distributed. When `runMode=local`, the sampler runs on the client node, and `HADOOP_CLIENT_OPTS` sets the heap size on the client node. When `runMode=distributed`,

Perfect Balance automatically sets the heap size for the sampler Hadoop job based on the `-Xmx` setting you provide in `HADOOP_CLIENT_OPTS`. Perfect Balance never changes the heap size for the map and reduce tasks of your job, only for its sampler job.

Using the Perfect Balance API

The `oracle.hadoop.balancer.Balancer` class contains methods for creating a partitioning plan, saving the plan to a file, and running the MapReduce job using the plan. You only need to add the code to the application's job driver Java class, not redesign the application. When you run a shell script to run the application, you can include Perfect Balance configuration settings.

Modifying Your Java Code to Use Perfect Balance

The Perfect Balance installation directory contains a complete example, including input data, of a Java MapReduce program that uses the Perfect Balance API.

For a description of the inverted index example and execution instructions, see `orabalancer-2.2.0-h2/examples/invindx/README.txt`.

To explore the modified Java code, see

`orabalancer-2.2.0-h2/examples/jsrc/oracle/hadoop/balancer/examples/invindx/InvertedIndexMapred.java` or `InvertedIndexMapreduce.java`.

The modifications to run Perfect Balance include the following:

- The `createBalancer` method validates the configuration properties and returns a `Balancer` instance.
- The `waitForCompletion` method samples the data and creates a partitioning plan.
- The `addBalancingPlan` method adds the partitioning plan to the job configuration settings.
- The `configureCountingReducer` method collects additional load statistics.
- The `save` method saves the partition report and generates the Job Analyzer report.

[Example 5-4](#) shows fragments from the inverted index Java code.

Example 5-4 Running Perfect Balance in a MapReduce Job

```
.
.
.
import oracle.hadoop.balancer.Balancer;
.
.
.
///// BEGIN: CODE TO INVOKE BALANCER (PART-1, before job submission) /////
Configuration conf = job.getConfiguration();

Balancer balancer = null;

boolean useBalancer =
    conf.getBoolean("oracle.hadoop.balancer.driver.balance", true);
if(useBalancer)
{
    balancer = Balancer.createBalancer(conf);
    balancer.waitForCompletion();
    balancer.addBalancingPlan(conf);
}
```

```
}

if(conf.getBoolean("oracle.hadoop.balancer.tools.useCountingReducer", true))
{
    Balancer.configureCountingReducer(conf);
}
////////// END: CODE TO INVOKE BALANCER (PART-1) //////////

boolean isSuccess = job.waitForCompletion(true);

//////////
// BEGIN: CODE TO INVOKE BALANCER (PART-2, after job completion, optional)
// If balancer ran, this saves the partition file report into the _balancer
// sub-directory of the job output directory. It also writes a JobAnalyzer
// report.
Balancer.save(job);
////////// END: CODE TO INVOKE BALANCER (PART-2) //////////
.
.
.
}
```

See Also: *Oracle Big Data Appliance Perfect Balance Java API Reference*

Running Your Modified Java Code with Perfect Balance

When you run your modified Java code, you can set the Perfect Balance properties by using the standard hadoop command syntax:

```
bin/hadoop jar application_jarfile.jar ApplicationClass \
-conf application_config.xml \
-conf perfect_balance_config.xml \
-D application_config_property \
-D perfect_balance_config_property \
-libjars application_jar_path.jar...
```

Example 5-5 runs a script named `pb_balanceapi.sh`, which runs the `InvertedIndexMapreduce` class example packaged in the Perfect Balance JAR file. The key load metric properties are set to the values recommended in the Job Analyzer report shown in [Figure 5-1](#).

To run the `InvertedIndexMapreduce` class example, see ["About the Perfect Balance Examples"](#) on page 5-17.

Example 5-5 *Running the InvertedIndexMapreduce Class*

```
$ cat pb_balanceapi.sh
BALANCER_HOME=/opt/oracle/orabalancer-2.2.0-h2
APP_JAR_FILE=/opt/oracle/orabalancer-2.2.0-h2/jlib/orabalancer-2.2.0.jar
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancer-2.2.0.jar:${BALANCER_
HOME}/jlib/commons-math-2.2.jar:$HADOOP_CLASSPATH
export HADOOP_USER_CLASSPATH_FIRST=true

hadoop jar ${APP_JAR_FILE}
oracle.hadoop.balancer.examples.invidx.InvertedIndexMapreduce \
-D mapreduce.input.fileinputformat.inputdir=invidx/input \
-D mapreduce.output.fileoutputformat.outputdir=jdoe_outdir_api \
-D mapreduce.job.name=jdoe_invidx_api \
-D mapreduce.job.reduces=10 \
-D oracle.hadoop.balancer.linearKeyLoad.keyWeight=93.981394 \
-D oracle.hadoop.balancer.linearKeyLoad.rowWeight=0.001126 \
```

```
-D oracle.hadoop.balancer.linearKeyLoad.byteWeight=0.0

$ sh ./balanceapi.sh
14/04/14 15:03:51 INFO balancer.Balancer: Creating balancer
14/04/14 15:03:51 INFO balancer.Balancer: Starting Balancer
14/04/14 15:03:51 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 15:03:54 INFO balancer.Balancer: Balancer completed
14/04/14 15:03:55 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 15:03:55 INFO mapreduce.JobSubmitter: number of splits:5
14/04/14 15:03:55 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_
1397066986369_3510
14/04/14 15:03:55 INFO impl.YarnClientImpl: Submitted application application_
1397066986369_3510
.
.
.
File Input Format Counters
Bytes Read=112652976
File Output Format Counters
Bytes Written=384974202
```

About the Perfect Balance Examples

The Perfect Balance installation files include a full set of examples that you can run immediately. The InvertedIndex example is a MapReduce application that creates an inverted index on an input set of text files. The inverted index maps words to the location of the words in the text files. The input data is included.

About the Examples in This Chapter

The InvertedIndex example provides the basis for all examples in this chapter. They use the same data set and run the same MapReduce application. The modifications to the InvertedIndex example simply highlight the steps you must perform in running your own applications with Perfect Balance.

If you want to run the examples in this chapter, or use them as the basis for running your own jobs, then make the following changes:

- If you are modifying the examples to run your own application, then add your application JAR files to HADOOP_CLASSPATH and `-libjars`.
- Ensure that the value of `mapreduce.input.fileinputformat.inputdir` identifies the location of your data.

The `invindx/input` directory contains the sample data for the InvertedIndex example. To use this data, you must first set it up. See ["Extracting the Example Data Set"](#) on page 5-18.

- Replace `jdoe` with your Hadoop user name.
- Set the `-conf` option to an existing configuration file.

The `jdoe_conf_invindx.xml` file is a modification of the configuration file for the InvertedIndex examples. The modified file does not have performance optimizing settings. You can use the example configuration file as is or modify it. See `/opt/oracle/orabalancer-2.2.0-h2/examples/invindx/conf_mapreduce.xml` (or `conf_mapred.xml`).

- Review the configuration settings in the file and in the shell script to ensure they are appropriate for your job.

- You can run the browser from your laptop or connect to Oracle Big Data Appliance using a client that supports graphical interfaces, such as VNC.

Extracting the Example Data Set

To run the InvertedIndex examples or any of the examples in this chapter, you must first set up the data files.

To extract the InvertedIndex data files:

1. Log in to a server where Perfect Balance is installed.
2. Change to the `examples/invindx` subdirectory:

```
cd /opt/oracle/orabalancer-2.2.0-h2/examples/invindx
```

3. Unzip the data and copy it to the HDFS `invindx/input` directory:

```
./invindx -setup
```

For complete instructions for running the InvertedIndex example, see `/opt/oracle/orabalancer-2.2.0-h2/examples/invindx/README.txt`.

Perfect Balance Configuration Property Reference

This section describes the Perfect Balance configuration properties and a few generic Hadoop MapReduce properties that Perfect Balance reads from the job configuration:

- [MapReduce Configuration Properties](#)
- [Job Analyzer Configuration Properties](#)
- [Perfect Balance Configuration Properties](#)

See "[About Configuring Perfect Balance](#)" on page 5-9 for a list of the properties organized into functional categories.

Note: CDH5 deprecates many MapReduce properties and replaces them with new properties. Perfect Balance continues to work with the old property names, but Oracle recommends that you use the new names. For the new MapReduce property names, see the Cloudera website at:

<http://archive.cloudera.com/cdh5/cdh/5/hadoop/hadoop-project-dist/hadoop-common/DeprecatedProperties.html>

MapReduce Configuration Properties

mapreduce.input.fileinputformat.inputdir

Type: String

Default Value: Not defined

Description: A comma-separated list of input directories.

mapreduce.inputformat.class

Type: String

Default Value: `org.apache.hadoop.mapreduce.lib.input.TextInputFormat`

Description: The full name of the `InputFormat` class.

mapreduce.map.class

Type: String

Default Value: `org.apache.hadoop.mapreduce.Mapper`

Description: The full name of the mapper class.

mapreduce.output.fileoutputformat.outputdir

Type: String

Default Value: Not defined

Description: The job output directory.

mapreduce.partition.class

Type: String

Default Value: `org.apache.hadoop.mapreduce.lib.partition.HashPartitioner`

Description: The full name of the partitioner class.

mapreduce.reduce.class

Type: String

Default Value: `org.apache.hadoop.mapreduce.Reducer`

Description: The full name of the reducer class.

Job Analyzer Configuration Properties**oracle.hadoop.balancer.application_id**

Type: String

Default Value: Not defined

Description: The job identifier of the job you want to analyze with Job Analyzer. This property is a parameter to the Job Analyzer utility in standalone mode on YARN clusters; it does not apply to MRv1 clusters. See ["Running Job Analyzer as a Standalone Utility"](#) on page 5-4.

oracle.hadoop.balancer.tools.jobConfPath

Type: String

Default Value: `${mapreduce.output.fileoutputformat.outputdir}/_logs/history`

Description: The path to a Hadoop job configuration file in MRv1 (not applicable to YARN). Job Analyzer uses this setting to locate the file.

oracle.hadoop.balancer.tools.jobHistoryPath

Type: String

Default Value: `${mapreduce.output.fileoutputformat.outputdir}/_logs/history`

Description: The path to a Hadoop job history file in MRv1 (not applicable to YARN). Job Analyzer uses this setting to locate the file.

oracle.hadoop.balancer.tools.writeKeyBytes

Type: Boolean

Default Value: `false`

Description: Controls whether the counting reducer collects the byte representations of the reduce keys for the Job Analyzer. Set this property to `true` to represent the unique key values in Base64 encoding in the report. A string representation of the key,

created using `key.toString`, is also provided in the report. This string value may not be unique for each key.

Perfect Balance Configuration Properties

oracle.hadoop.balancer.autoAnalyze

Type: Enumeration

Default Value: BASIC_REPORT if `oracle.hadoop.balancer.autoBalance` is true; otherwise NONE

Description: Controls the behavior of the Job Analyzer when it is called using Perfect Balance. The following values are valid:

- NONE: Disables Job Analyzer.
- BASIC_REPORT: Enables Job Analyzer
- REDUCER_REPORT: Enables Job Analyzer such that it collects additional load statistics for each reduce task in a job. See ["Collecting Additional Metrics"](#) on page 5-7..

Perfect Balance uses this property; the Perfect BalanceAPI does not.

oracle.hadoop.balancer.autoBalance

Type: Boolean

Default Value: false

Description: Controls whether load balancing is enabled. Set to false to turn off balancing. Perfect Balance uses this property; the Perfect Balance API does not.

oracle.hadoop.balancer.confidence

Type: Float

Default Value: 0.95

Description: The statistical confidence indicator for the load factor specified by the `oracle.hadoop.balancer.maxLoadFactor` property.

This property accepts values greater than or equal to 0.5 and less than 1.0 ($0.5 \leq \text{value} < 1.0$). A value less than 0.5 resets the property to its default value. Oracle recommends a value greater than or equal to 0.9. Typical values are 0.95 and 0.99.

oracle.hadoop.balancer.enableSorting

Type: Boolean

Default Value: false

Description: Controls how the map output keys are **chopped**, that is, split into smaller keys:

- false: Uses a hash function.
- true: Uses the map output key sorting comparator as a total-order partitioning function. The balancer preserves the total order over the values of the chopped keys.

Perfect Balance sets this property to false when the job is not configured for secondary sorting or when `oracle.hadoop.balancer.keyLoad.minChopBytes` is -1.

oracle.hadoop.balancer.inputFormat.mapred.map.tasks

Type: Integer

Default Value: 100

Description: Sets the Hadoop `mapred.map.tasks` property for the duration of sampling, just before calling the input format `getSplits` method. It does not change `mapred.map.tasks` for the actual job. The optimal number of map tasks is a trade-off between obtaining a good sample (larger number) and having finite memory resources (smaller number).

Set this property to a value greater than or equal to one (1). A value less than 1 disables the property.

Some input formats, such as `DBInputFormat`, use this property as a hint to determine the number of splits returned by `getSplits`. Higher values indicate that more chunks of data are sampled at random, which improves the sample.

You can increase the value for larger data sets, that is, more than a million rows of about 100 bytes per row. However, extremely large values can cause the input format's `getSplits` method to run out of memory by returning too many splits.

oracle.hadoop.balancer.inputFormat.mapred.max.split.size

Type: Long

Default Value: 1048576 (1 MB)

Description: Sets the Hadoop `mapred.max.split.size` property for the duration of sampling, just before calling the input format's `getSplits` method. It does not change `mapred.max.split.size` for the actual job.

Set this property to a value greater than or equal to one (1). A value less than 1 disables the property. The optimal split size is a trade-off between obtaining a good sample (smaller splits) and efficient I/O performance (larger splits).

Some input formats, such as `FileInputFormat`, use the maximum split size as a hint to determine the number of splits returned by `getSplits`. Smaller split sizes indicate that more chunks of data are sampled at random, which improves the sample. Set the value small enough for good sampling performance, but no smaller. Extremely small values can cause inefficient I/O performance, while not improving the sample.

You can increase the value for larger data sets (tens of terabytes) or if the input format's `getSplits` method throws an out of memory error. Large splits are better for I/O performance, but not for sampling.

oracle.hadoop.balancer.keyLoad.minChopBytes

Type: Long

Default Value: 0

Description: Controls whether Perfect Balance chops large map output keys into medium keys:

- -1: Perfect Balance does not chop large map output keys.
- 0: Perfect Balance chops large map output keys and determines the optimal size of each medium key.
- *Positive integer*: Perfect Balance chops large map output keys into medium keys with a size greater than or equal to the specified integer.

oracle.hadoop.balancer.linearKeyLoad.byteWeight

Type: Float

Default Value: 0.05

Description: Weights the number of bytes per key in the linear key load model specified by the `oracle.hadoop.balancer.KeyLoadLinear` class.

oracle.hadoop.balancer.linearKeyLoad.feedbackDir

Type: String

Default Value: Not defined

Description: The path to a directory that contains the Job Analyzer report for a job that it previously analyzed. The sampler reads this report for feedback to use to optimize the current balancing plan. You can set this property to the Job Analyzer report directory of a job that is the same or similar to the current job, so that the feedback is directly applicable.

If the feedback directory contains a Job Analyzer report with recommended values for the Perfect Balance linear key load model coefficients, then Perfect Balance automatically reads and uses them. The recommended values take precedence over user-specified values in these configuration parameters:

- `oracle.hadoop.balancer.linearKeyLoad.byteWeight`
- `oracle.hadoop.balancer.linearKeyLoad.keyWeight`
- `oracle.hadoop.balancer.linearKeyLoad.rowWeight`

Job Analyzer attempts to recommend good values for these coefficients. However, Perfect Balance reads the load model coefficients from this list of configuration properties under the following circumstances:

- The `feedbackDir` property is not set.
- The `feedbackDir` property is set, but the Job Analyzer report in the specified directory does not contain a good recommendation for the load model coefficients.

oracle.hadoop.balancer.linearKeyLoad.keyWeight

Type: Float

Default Value: 50.0

Description: Weights the number of medium keys per large key in the linear key load model specified by the `oracle.hadoop.balancer.KeyLoadLinear` class.

oracle.hadoop.balancer.linearKeyLoad.rowWeight

Type: Float

Default Value: 0.05

Description: Weights the number of rows per key in the linear key load model specified by the `oracle.hadoop.balancer.KeyLoadLinear` class.

oracle.hadoop.balancer.maxLoadFactor

Type: Float

Default Value: 0.05

Description: The target reducer load factor that you want the balancer's partition plan to achieve.

The load factor is the relative deviation from an estimated value. For example, if `maxLoadFactor=0.05` and `confidence=0.95`, then with a confidence greater than 95%, the job's reducer loads should be, at most, 5% greater than the value in the partition plan.

The values of these two properties determine the sampler's stopping condition. The balancer samples until it can generate a plan that guarantees the specified load factor at the specified confidence level. This guarantee may not hold if the sampler stops early because of other stopping conditions, such as the number of samples exceeds [oracle.hadoop.balancer.maxSamplesPct](#). The partition report logs the stopping condition.

See [oracle.hadoop.balancer.confidence](#).

oracle.hadoop.balancer.maxSamplesPct

Type: Float

Default Value: 0.01 (1%)

Description: Limits the number of samples that Perfect Balance can collect to a fraction of the total input records. A value less than zero disables the property (no limit).

You may need to increase the value for Hadoop applications with very unbalanced reducer partitions or densely clustered map-output keys. The sampler needs to sample more data to achieve a good partitioning plan in these cases.

See [oracle.hadoop.balancer.useClusterStats](#).

oracle.hadoop.balancer.minSplits

Type: Integer

Default Value: 5

Description: Sets the minimum number of splits that the sampler reads. If the total number of splits is less than this value, then the sampler reads all splits. Set this property to a value greater than or equal to one (1). A nonpositive number sets the property to 1.

oracle.hadoop.balancer.numThreads

Type: Integer

Default Value: 5

Description: Number of sampler threads. Set this value based on the processor and memory resources available on the node where the job is initiated. A higher number of sampler threads implies higher concurrency in sampling. Set this property to one (1) to disable multithreading in the sampler.

oracle.hadoop.balancer.report.override

Type: Boolean

Default Value: false

Description: Controls whether Perfect Balance overwrites files in the location specified by the [oracle.hadoop.balancer.reportPath](#) property. By default, Perfect Balance does not overwrite files; it throws an exception. Set this property to true to allow partition reports to be overwritten.

oracle.hadoop.balancer.reportPath

Type: String

Default Value: *directory/orabalancer_report-random_unique_string.xml*, where *directory* for HDFS is the home directory of the user who submits the job. For the local file system, it is the directory where the job is submitted.

Description: The path where Perfect Balance writes the partition report before the Hadoop job output directory is available, that is, before the MapReduce job finishes

running. At the end of the job, Perfect Balance moves the file to *job_output_dir/_balancer/orabalancer_report.xml*. In the API, the save method does this task.

oracle.hadoop.balancer.runMode

Type: String

Default Value: `local`

Description: Specifies how to run the Perfect Balance sampler. The following values are valid:

- `local`: The sampler runs on the client node where the job is submitted.
- `distributed`: The sampler runs as a Hadoop job. If the job uses the distributed cache, then Perfect Balance automatically sets this property to `distributed`.

If this property is set to an invalid string, Perfect Balance resets it to `local`.

oracle.hadoop.balancer.tmpDir

Type: String

Default Value: `/tmp/orabalancer-user_name`

Description: The path to a staging directory in the file system of the job output directory (HDFS or local). Perfect Balance creates the directory if it does not exist, and copies the partition report to it for loading into the Hadoop distributed cache.

oracle.hadoop.balancer.useClusterStats

Type: Boolean

Default Value: `true`

Description: Enables the sampler to use cluster sampling statistics. These statistics improve the accuracy of sampled estimates, such as the number of records in a map-output key, when the map-output keys are distributed in clusters across input splits, instead of being distributed independently across all input splits.

Set this property to `false` only if you are absolutely certain that the map-output keys are not clustered. This setting improves the sampler's estimates only when there is, in fact, no clustering. Oracle recommends leaving this property set to `true`, because the distribution of map-output keys is usually unknown.

oracle.hadoop.balancer.useMapreduceApi

Type: Boolean

Default Value: `true`

Description: Identifies the MapReduce API used in the Hadoop job:

- `true`: The job uses the mapreduce API.
- `false`: The job uses the mapred API.

Part III

Oracle Big Data SQL

This section describes the software distributed under the Oracle Big Data SQL license. It contains the following chapters:

- [Chapter 6, "Using Oracle Big Data SQL for Data Access"](#)
- [Chapter 7, "Oracle Big Data SQL Reference"](#)
- [Chapter 8, "Copying Oracle Tables to Hadoop"](#)

Using Oracle Big Data SQL for Data Access

This chapter describes Oracle Big Data SQL. It contains the following topics:

- [What Is Oracle Big Data SQL?](#)
- [Installing Oracle Big Data SQL](#)
- [Creating an Oracle External Table for Hive Data](#)
- [Creating an Oracle External Table for Oracle NoSQL Database](#)
- [Creating an Oracle External Table for Apache HBase](#)
- [Creating an Oracle External Table for HDFS Files](#)
- [About the SQL CREATE TABLE Statement](#)
- [About Data Type Conversions](#)
- [Querying External Tables](#)
- [About Oracle Big Data SQL on Oracle Exadata Database Machine](#)

What Is Oracle Big Data SQL?

Oracle Big Data SQL supports queries against vast amounts of big data stored in multiple data sources, including Apache Hive, HDFS, Oracle NoSQL Database, and Apache HBase. You can view and analyze data from various data stores together, as if it were all stored in an Oracle database.

Using Oracle Big Data SQL, you can query data stored in a Hadoop cluster using the complete SQL syntax. You can execute the most complex SQL `SELECT` statements against data in Hadoop, either manually or using your existing applications, to tease out the most significant insights. For example, users of the Oracle Advanced Analytics database option can apply their data mining models, which reside in Oracle Database, to data that is resident on Oracle Big Data Appliance.

The following sections provide further details:

- [About Oracle External Tables](#)
- [About the Access Drivers for Oracle Big Data SQL](#)
- [About Smart Scan Technology](#)
- [About Data Security with Oracle Big Data SQL](#)

About Oracle External Tables

Oracle Big Data SQL provides external tables with next generation performance gains. An **external table** is an Oracle Database object that identifies and describes the location of data outside of a database. You can query an external table using the same SQL `SELECT` syntax that you use for any other database tables.

External tables use **access drivers** to parse the data outside the database. Each type of external data requires a unique access driver. This release of Oracle Big Data SQL includes two access drivers for big data: one for accessing data stored in Apache Hive, and the other for accessing data stored in Hadoop Distributed File System (HDFS) files.

About the Access Drivers for Oracle Big Data SQL

By querying external tables, you can access data stored in HDFS and Hive tables as if that data was stored in tables in an Oracle database. Oracle Database accesses the data by using the metadata provided when the external table was created.

Oracle Database 12.1.0.2 supports two new access drivers for Oracle Big Data SQL:

- `ORACLE_HIVE`: Enables you to create Oracle external tables over Apache Hive data sources. Use this access driver when you already have Hive tables defined for your HDFS data sources. `ORACLE_HIVE` can also access data stored in other locations, such as HBase, that have Hive tables defined for them.
- `ORACLE_HDFS`: Enables you to create Oracle external tables directly over files stored in HDFS. This access driver uses Hive syntax to describe a data source, assigning default column names of `COL_1`, `COL_2`, and so forth. You do not need to create a Hive table manually as a separate step.

Instead of acquiring the metadata from a Hive metadata store the way that `ORACLE_HIVE` does, the `ORACLE_HDFS` access driver acquires all of the necessary information from the access parameters. The `ORACLE_HDFS` access parameters are required to specify the metadata, and are stored as part of the external table definition in Oracle Database.

Oracle Big Data SQL uses these access drivers to optimize query performance.

About Smart Scan Technology

External tables do not have traditional indexes, so that queries against them typically require a full table scan. However, Oracle Big Data SQL extends SmartScan capabilities, such as filter-predicate offloads, to Oracle external tables with the installation of Exadata storage server software on Oracle Big Data Appliance. This technology enables Oracle Big Data Appliance to discard a huge portion of irrelevant data—up to 99 percent of the total—and return much smaller result sets to Oracle Exadata Database Machine. End users obtain the results of their queries significantly faster, as the direct result of a reduced load on Oracle Database and reduced traffic on the network.

See Also: *Oracle Database Concepts* for a general introduction to external tables and pointers to more detailed information in the Oracle Database documentation library

About Data Security with Oracle Big Data SQL

Oracle Big Data Appliance already provides numerous security features to protect data stored in a CDH cluster on Oracle Big Data Appliance:

- **Kerberos authentication:** Requires users and client software to provide credentials before accessing the cluster.
- **Apache Sentry authorization:** Provides fine-grained, role-based authorization to data and metadata.
- **On-disk encryption:** Protects the data on disk and at rest. For normal user access, the data is automatically decrypted.
- **Oracle Audit Vault and Database Firewall monitoring:** The Audit Vault plug-in on Oracle Big Data Appliance collects audit and logging data from MapReduce, HDFS, and Oozie services. You can then use Audit Vault Server to monitor these services on Oracle Big Data Appliance

Oracle Big Data SQL adds the full range of Oracle Database security features to this list. You can apply the same security policies and rules to your Hadoop data that you apply to your relational data.

Installing Oracle Big Data SQL

Oracle Big Data SQL is available only on Oracle Exadata Database Machine connected to Oracle Big Data Appliance. You must install the Oracle Big Data SQL software on both systems.

The following topics explain how to install Oracle Big Data SQL:

- [Prerequisites for Using Oracle Big Data SQL 1.1](#)
- [Performing the Installation](#)
- [Running the Post-Installation Script for Oracle Big Data SQL](#)

Prerequisites for Using Oracle Big Data SQL 1.1

Oracle Exadata Database Machine must comply with the following requirements:

- Compute servers run Oracle Database and Oracle Enterprise Manager Grid Control 12.1.0.2.1 or later.
- Storage servers run Exadata storage server software 12.1.1.1 or 12.1.1.0.
- Oracle Exadata Database Machine is configured on the same InfiniBand subnet as Oracle Big Data Appliance
- Oracle Exadata Database Machine is connected to Oracle Big Data Appliance by the InfiniBand network.

Performing the Installation

Take these steps to install the Oracle Big Data SQL software on Oracle Big Data Appliance and Oracle Exadata Database Machine:

1. Download the Oracle Database one-off patch to 12.1.0.2.1.
2. On all Oracle Exadata Database Machine compute servers, install the patch on:
 - Grid Infrastructure home
 - Oracle Database homes

See the patch `README` for step-by-step instructions for installing the patch.

3. On Oracle Big Data Appliance, install or upgrade the software to the latest version. See *Oracle Big Data Appliance Owner's Guide*.

You can select Oracle Big Data SQL as an installation option when using the Oracle Big Data Appliance Configuration Generation Utility. See *Oracle Big Data Appliance Owner's Guide*.

4. If Oracle Big Data SQL is not enabled during the installation, then use the `bdaccli` utility:

```
# bdaccli enable big_data_sql
```

See *Oracle Big Data Appliance Owner's Guide*.

5. On Oracle Exadata Database Machine, run the post-installation script.

See ["Running the Post-Installation Script for Oracle Big Data SQL"](#) on page 6-4.

You can use Cloudera Manager to verify that Oracle Big Data SQL is up and running. See ["Managing Oracle Big Data SQL"](#) on page 2-24.

Running the Post-Installation Script for Oracle Big Data SQL

To run the Oracle Big Data SQL post-installation script:

1. On Oracle Exadata Database Machine, ensure that the Oracle Database listener is running and listening on an interprocess communication (IPC) interface.
2. Verify the name of the Oracle installation owner. Typically, the `oracle` user owns the installation.
3. Verify that the same user name (such as `oracle`) exists on Oracle Big Data Appliance.
4. Download the `bds-exa-install.sh` installation script from the node where Mammoth is installed, typically the first node in the cluster. You can use a command such as `wget` or `curl`. This example copies the script from `bda1node07`:

```
wget http://bda1node07/bda/bds-exa-install.sh
```

5. As root, run the script and pass it the system identifier (SID). In this example, the SID is `orcl`:

```
./bds-exa-install.sh oracle_sid=orcl
```

Note: If the Oracle installation owner is not `oracle`, then use the `--install-user` option. See ["Running the bds-exa-install Script"](#) on page 6-4.

6. Repeat step 5 for each database instance.

When the script completes, Oracle Big Data SQL is running on the database instance. However, if events cause the Oracle Big Data SQL agent to stop, then you must restart it. See ["Starting and Stopping the Big Data SQL Agent"](#) on page 6-20.

Running the bds-exa-install Script

The `bds-exa-install` script generates a custom installation script that is run by the owner of the Oracle home directory. That secondary script installs all the files need by Oracle Big Data SQL into the `$ORACLE_HOME/bigdatasql` directory. For Oracle NoSQL Database support, it installs the client library (`kvclient.jar`). It also creates the database directory objects, and the database links for the multithreaded Oracle Big Data SQL agent.

If the operating system user who owns Oracle home is not named `oracle`, then use the `--install-user` option to specify the owner.

Alternatively, you can use the `--generate-only` option to create the secondary script, and then run it as the owner of `$ORACLE_HOME`.

bds-ex-install Syntax

The following is the `bds-exa-install` syntax:

```
./bds-exa-install.sh oracle_sid=name [option]
```

The option names are preceded by two hyphens (`--`):

--generate-only={true | false}

Set to `true` to generate the secondary script, but not run it, or `false` to generate and run it in one step (default).

--install-user=user_name

The operating system user who owns the Oracle Database installation. The default value is `oracle`.

Creating an Oracle External Table for Hive Data

You can easily create an Oracle external table for data in Apache Hive. Because the metadata is available to Oracle Database, you can query the data dictionary for information about Hive tables. Then you can use a PL/SQL function to generate a basic SQL `CREATE TABLE EXTERNAL ORGANIZATION` statement. You can modify the statement before execution to customize the external table.

Obtaining Information About a Hive Table

The `DBMS_HADOOP` PL/SQL package contains a function named `CREATE_EXTDDL_FOR_HIVE`. It returns the data dictionary language (DDL) to create an external table for accessing a Hive table. This function requires you to provide basic information about the Hive table:

- Name of the Hadoop cluster
- Name of the Hive database
- Name of the Hive table
- Whether the Hive table is partitioned

You can obtain this information by querying the `ALL_HIVE_TABLES` data dictionary view. It displays information about all Hive tables that you can access from Oracle Database.

This example shows that the current user has access to an unpartitioned Hive table named `RATINGS_HIVE_TABLE` in the default database. A user named `JDOE` is the owner.

```
SQL> SELECT cluster_id, database_name, owner, table_name, partitioned FROM all_hive_tables;
```

CLUSTER_ID	DATABASE_NAME	OWNER	TABLE_NAME	PARTITIONED
hadoop1	default	jdoe	ratings_hive_table	UN-PARTITIONED

See Also: ["Static Data Dictionary Views for Hive"](#) on page 7-23

Using the CREATE_EXTDDL_FOR_HIVE Function

With the information from the data dictionary, you can use the `CREATE_EXTDDL_FOR_HIVE` function of `DBMS_HADOOP`. This example specifies a database table name of `RATINGS_DB_TABLE` in the current schema. The function returns the text of the `CREATE TABLE` command in a local variable named `DDLout`, but does not execute it.

```
DECLARE
    DDLout VARCHAR2(4000);
BEGIN
    dbms_hadoop.create_extddl_for_hive(
        CLUSTER_ID=>'hadoop1',
        DB_NAME=>'default',
        HIVE_TABLE_NAME=>'ratings_hive_table',
        HIVE_PARTITION=>FALSE,
        TABLE_NAME=>'ratings_db_table',
        PERFORM_DDL=>FALSE,
        TEXT_OF_DDL=>DDLout
    );
    dbms_output.put_line(DDLout);
END;
/
```

When this procedure runs, the `PUT_LINE` function displays the `CREATE TABLE` command:

```
CREATE TABLE ratings_db_table (
    c0 VARCHAR2(4000),
    c1 VARCHAR2(4000),
    c2 VARCHAR2(4000),
    c3 VARCHAR2(4000),
    c4 VARCHAR2(4000),
    c5 VARCHAR2(4000),
    c6 VARCHAR2(4000),
    c7 VARCHAR2(4000))
ORGANIZATION EXTERNAL
  (TYPE ORACLE_HIVE DEFAULT DIRECTORY DEFAULT_DIR
   ACCESS PARAMETERS
     (
       com.oracle.bigdata.cluster=hadoop1
       com.oracle.bigdata.tablename=default.ratings_hive_table
     )
   ) PARALLEL 2 REJECT LIMIT UNLIMITED
```

You can capture this information in a SQL script, and use the access parameters to change the Oracle table name, the column names, and the data types as desired before executing it. You might also use access parameters to specify a date format mask.

The `ALL_HIVE_COLUMNS` view shows how the default column names and data types are derived. This example shows that the Hive column names are `C0` to `C7`, and that the Hive `STRING` data type maps to `VARCHAR2(4000)`:

```
SQL> SELECT table_name, column_name, hive_column_type, oracle_column_type FROM
all_hive_columns;
```

TABLE_NAME	COLUMN_NAME	HIVE_COLUMN_TYPE	ORACLE_COLUMN_TYPE
ratings_hive_table	c0	string	VARCHAR2(4000)
ratings_hive_table	c1	string	VARCHAR2(4000)
ratings_hive_table	c2	string	VARCHAR2(4000)
ratings_hive_table	c3	string	VARCHAR2(4000)

```

ratings_hive_table  c4          string      VARCHAR2(4000)
ratings_hive_table  c5          string      VARCHAR2(4000)
ratings_hive_table  c6          string      VARCHAR2(4000)
ratings_hive_table  c7          string      VARCHAR2(4000)

```

8 rows selected.

See Also: ["DBMS_HADOOP PL/SQL Package"](#) on page 7-2

Developing a CREATE TABLE Statement for ORACLE_HIVE

You can choose between using DBMS_HADOOP and developing a CREATE TABLE statement from scratch. In either case, you may need to set some access parameters to modify the default behavior of ORACLE_HIVE.

Using the Default ORACLE_HIVE Settings

The following statement creates an external table named ORDER to access Hive data:

```

CREATE TABLE order (cust_num    VARCHAR2(10),
                     order_num   VARCHAR2(20),
                     description VARCHAR2(100),
                     order_total NUMBER (8,2))
  ORGANIZATION EXTERNAL (TYPE oracle_hive);

```

Because no access parameters are set in the statement, the ORACLE_HIVE access driver uses the default settings to do the following:

- Connects to the default Hadoop cluster.
- Uses a Hive table named order. An error results if the Hive table does not have fields named CUST_NUM, ORDER_NUM, DESCRIPTION, and ORDER_TOTAL.
- Sets the value of a field to NULL if there is a conversion error, such as a CUST_NUM value longer than 10 bytes.

Overriding the Default ORACLE_HIVE Settings

You can set properties in the ACCESS PARAMETERS clause of the external table clause, which override the default behavior of the access driver. The following clause includes the com.oracle.bigdata.overflow access parameter. When this clause is used in the previous example, it truncates the data for the DESCRIPTION column that is longer than 100 characters, instead of throwing an error:

```

(TYPE oracle_hive
 ACCESS PARAMETERS (
   com.oracle.bigdata.overflow={"action":"truncate", "col":"DESCRIPTION"} ))

```

The next example sets most of the available parameters for ORACLE_HIVE:

```

CREATE TABLE order (cust_num VARCHAR2(10),
                     order_num VARCHAR2(20),
                     order_date DATE,
                     item_cnt NUMBER,
                     description VARCHAR2(100),
                     order_total (NUMBER(8,2)) ORGANIZATION EXTERNAL
  (TYPE oracle_hive
   ACCESS PARAMETERS (
     com.oracle.bigdata.tablename: order_db.order_summary
     com.oracle.bigdata.colmap:    {"col":"ITEM_CNT", \
                                   "field":"order_line_item_count"}

```

```

com.oracle.bigdata.overflow: {"action":"TRUNCATE", \
                             "col":"DESCRIPTION"}
com.oracle.bigdata.erroropt: [{ "action":"replace", \
                              "value":"INVALID_NUM" , \
                              "col":["CUST_NUM", "ORDER_NUM"]} , \
                             { "action":"reject", \
                              "col":"ORDER_TOTAL"}
))

```

The parameters make the following changes in the way that the ORACLE_HIVE access driver locates the data and handles error conditions:

- `com.oracle.bigdata.tablename`: Handles differences in table names. ORACLE_HIVE looks for a Hive table named `ORDER_SUMMARY` in the `ORDER.DB` database.
- `com.oracle.bigdata.colmap`: Handles differences in column names. The Hive `ORDER_LINE_ITEM_COUNT` field maps to the Oracle `ITEM_CNT` column.
- `com.oracle.bigdata.overflow`: Truncates string data. Values longer than 100 characters for the `DESCRIPTION` column are truncated.
- `com.oracle.bigdata.erroropt`: Replaces bad data. Errors in the data for `CUST_NUM` or `ORDER_NUM` set the value to `INVALID_NUM`.

Creating an Oracle External Table for Oracle NoSQL Database

You can use the ORACLE_HIVE access driver to access data stored in Oracle NoSQL Database. However, you must first create a Hive external table that accesses the KVStore. Then you can create an external table in Oracle Database over it, similar to the process described in ["Creating an Oracle External Table for Hive Data"](#) on page 6-5.

This section contains the following topics:

- [Creating a Hive External Table for Oracle NoSQL Database](#)
- [Creating the Oracle Database Table for Oracle NoSQL Data](#)
- [About Column Data Type Mappings](#)
- [Example of Accessing Data in Oracle NoSQL Database](#)

Creating a Hive External Table for Oracle NoSQL Database

To provide access to the data in Oracle NoSQL Database, you create a Hive external table over the Oracle NoSQL table. Oracle Big Data SQL provides a storage handler named `oracle.kv.hadoop.hive.table.TableStorageHandler` that enables Hive to read the Oracle NoSQL Database table format.

The following is the basic syntax of a Hive `CREATE TABLE` statement for a Hive external table over an Oracle NoSQL table:

```

CREATE EXTERNAL TABLE tablename colname coltype [, colname coltype, ...]
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
    "oracle.kv.kvstore" = "database",
    "oracle.kv.hosts" = "nosql_node1:port[, nosql_node2:port...]",
    "oracle.kv.hadoop.hosts" = "hadoop_node1[,hadoop_node2...]",
    "oracle.kv.tableName" = "table_name");

```


Hive CREATE TABLE Parameters

tablename

The name of the Hive external table being created.

This table name will be used in SQL queries issued in Oracle Database, so choose a name that is appropriate for users. The name of the external table that you create in Oracle Database must be identical to the name of this Hive table.

Table, column, and field names are case insensitive in Oracle NoSQL Database, Apache Hive, and Oracle Database.

colname coltype

The names and data types of the columns in the Hive external table. See [Table 6-1](#) for the data type mappings between Oracle NoSQL Database and Hive.

Hive CREATE TABLE TBLPROPERTIES Clause

oracle.kv.kvstore

The name of the KVStore. Only upper- and lowercase letters and digits are valid in the name.

oracle.kv.hosts

A comma-delimited list of host names and port numbers in the Oracle NoSQL Database cluster. Each string has the format *hostname:port*. Enter multiple names to provide redundancy in the event that a host fails.

oracle.kv.hadoop.hosts

A comma-delimited list of all host names in the CDH cluster in Oracle Big Data Appliance with Oracle Big Data SQL enabled.

oracle.kv.tableName

The name of the table in Oracle NoSQL Database that stores the data for this Hive external table.

See Also: *Apache Hive Language Manual DDL* at

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/TruncateTable>

Creating the Oracle Database Table for Oracle NoSQL Data

Use the following syntax to create an external table in Oracle Database that can access the Oracle NoSQL data through a Hive external table:

```
CREATE TABLE tablename(colname colType[, colname colType...])
  ORGANIZATION EXTERNAL
  (TYPE ORACLE_HIVE DEFAULT DIRECTORY directory
  ACCESS PARAMETERS
    (access parameters)
  )
  REJECT LIMIT UNLIMITED;
```

In this syntax, you identify the column names and data types. For more about this syntax, see ["About the SQL CREATE TABLE Statement"](#) on page 6-16.

About Column Data Type Mappings

When Oracle Big Data SQL retrieves data from Oracle NoSQL Database, the data is converted twice to another data type:

- To a Hive data type when the data is read into the columns of the Hive external table.
- To an Oracle data type when the data is read into the columns of an Oracle Database external table.

[Table 6–1](#) identifies the supported Oracle NoSQL data types and their mappings to Hive and Oracle Database data types. Oracle Big Data SQL does not support the Oracle NoSQL complex data types Array, Map, and Record.

Table 6–1 Oracle NoSQL Database Data Type Mappings

Oracle NoSQL Database Data Type	Apache Hive Data Type	Oracle Database Data Type
String	STRING	VARCHAR2
Boolean	BOOLEAN	NUMBER ¹
Integer	INT	NUMBER
Long	INT	NUMBER
Double	DOUBLE	NUMBER(<i>p,s</i>)
Float	FLOAT	NUMBER(<i>p,s</i>)

¹ 0 for false, and 1 for true

Example of Accessing Data in Oracle NoSQL Database

This example uses the sample data provided with the Oracle NoSQL Database software:

- [Creating the Oracle NoSQL Database Example Table](#)
- [Creating the Example Hive Table for vehicleTable](#)
- [Creating the Oracle Table for VEHICLES](#)

Creating the Oracle NoSQL Database Example Table

Verify that the following files reside in the `examples/hadoop/table` directory:

```
create_vehicle_table.kvs
CountTableRows.java
LoadVehicleTable.java
```

This example runs on a Oracle Big Data Appliance server named `bda1node07` and uses a KVStore named `BDAKV`.

To create and populate the sample table in Oracle NoSQL Database:

1. Open a connection to an Oracle NoSQL Database node on Oracle Big Data Appliance.
2. Create a table named `vehicleTable`. The following example uses the `load` command to run the commands in `create_vehicle_table.kvs`:

```
$ cd NOSQL_HOME
$ java -jar lib/kvcli.jar -host bda1node07 -port 5000 \
  load -file examples/hadoop/table/create_vehicle_table.kvs
```

3. Compile LoadVehicleTable.java:

```
$ javac -cp examples:lib/kvclient.jar
examples/hadoop/table/LoadVehicleTable.java
```

4. Execute the LoadVehicleTable class to populate the table:

```
$ java -cp examples:lib/kvclient.jar hadoop.table.LoadVehicleTable -host
bda1node07 -port 5000 -store BDAKV
{"type":"auto","make":"Chrysler","model":"PTCruiser","class":"4WheelDrive","col
o
r":"white","price":20743.240234375,"count":30}
{"type":"suv","make":"Ford","model":"Escape","class":"FrontWheelDrive","color":
"
.
.
.
10 new records added
```

The vehicleTable table contains the following fields:

Field Name	Data Type
type	STRING
make	STRING
model	STRING
class	STRING
color	STRING
price	DOUBLE
count	INTEGER

Creating the Example Hive Table for vehicleTable

The following example creates a Hive table named `VEHICLES` that accesses `vehicleTable` in the BDAKV KVStore. Oracle Big Data Appliance is configured with a CDH cluster in the first six servers (bda1node01 to bda1node06) and an Oracle NoSQL Database cluster in the next three servers (bda1node07 to bda1node09).

```
CREATE EXTERNAL TABLE IF NOT EXISTS vehicles
(
  type STRING,
  make STRING,
  model STRING,
  class STRING,
  color STRING,
  price DOUBLE,
  count INT)
COMMENT 'Accesses data in vehicleTable in the BDAKV KVStore'
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES
(
  "oracle.kv.kvstore" = "BDAKV",
  "oracle.kv.hosts" = "bda1node07.example.com:5000,bda1node08.example.com:5000",
  "oracle.kv.hadoop.hosts" =
  "bda1node01.example.com,bda1node02.example.com,bda1node03.example.com,bda1node04.e
  xample.com,bda1node05.example.com,bda1node06.example.com",
  "oracle.kv.tableName" = "vehicleTable");
```

The `DESCRIBE` command lists the columns in the `VEHICLES` table:

```
hive> DESCRIBE vehicles;
OK
type                string                from deserializer
make                 string                from deserializer
model                string                from deserializer
class                string                from deserializer
color                string                from deserializer
price                double               from deserializer
count                int                  from deserializer
```

A query against the Hive VEHICLES table returns data from the Oracle NoSQL vehicleTable table:

```
hive> SELECT make, model, class
      FROM vehicletable
      WHERE type='truck' AND color='red'
      ORDER BY make, model;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
.
.
.
Chrysler    Ram1500      RearWheelDrive
Chrysler    Ram2500      FrontWheelDrive
Ford        F150         FrontWheelDrive
Ford        F250         RearWheelDrive
Ford        F250         AllWheelDrive
Ford        F350         RearWheelDrive
GM          Sierra       AllWheelDrive
GM          Silverado1500 RearWheelDrive
GM          Silverado1500 AllWheelDrive
GM          Silverado1500 4WheelDrive
```

Creating the Oracle Table for VEHICLES

After you create the Hive table, the metadata is available in the Oracle Database static data dictionary views. The following SQL `SELECT` statement returns information about the Hive table created in the previous topic:

```
SQL> SELECT table_name, column_name, hive_column_type
      FROM all_hive_columns
      WHERE table_name='vehicles';
```

TABLE_NAME	COLUMN_NAME	HIVE_COLUMN_TYPE
vehicles	type	string
vehicles	make	string
vehicles	model	string
vehicles	class	string
vehicles	color	string
vehicles	price	double
vehicles	count	int

The next SQL `CREATE TABLE` statement generates an external table named `VEHICLES` over the Hive `VEHICLES` table, using the `ORACLE_HIVE` access driver. The name of the table in Oracle Database must be identical to the name of the table in Hive. However, both Oracle NoSQL Database and Oracle Database are case insensitive.

```
CREATE TABLE vehicles
  (type VARCHAR2(10), make VARCHAR2(12), model VARCHAR2(20),
```

```

class VARCHAR2(40), color VARCHAR2(20), price NUMBER(8,2),
count NUMBER)
ORGANIZATION EXTERNAL
  (TYPE ORACLE_HIVE DEFAULT DIRECTORY DEFAULT_DIR
   ACCESS PARAMETERS
     (com.oracle.bigdata.debug=true com.oracle.bigdata.log.opt=normal))
  REJECT LIMIT UNLIMITED;

```

This SQL `SELECT` statement retrieves all rows for red trucks from `vehicleTable` in Oracle NoSQL Database:

```

SQL> SELECT make, model, class
      FROM vehicles
      WHERE type='truck' AND color='red'
      ORDER BY make, model;

```

MAKE	MODEL	CLASS
Chrysler	Ram1500	RearWheelDrive
Chrysler	Ram2500	FrontWheelDrive
Ford	F150	FrontWheelDrive
Ford	F250	AllWheelDrive
Ford	F250	RearWheelDrive
Ford	F350	RearWheelDrive
GM	Sierra	AllWheelDrive
GM	Silverado1500	RearWheelDrive
GM	Silverado1500	4WheelDrive
GM	Silverado1500	AllWheelDrive

Creating an Oracle External Table for Apache HBase

You can also use the `ORACLE_HIVE` access driver to access data stored in Apache HBase. However, you must first create a Hive external table that accesses the HBase table. Then you can create an external table in Oracle Database over it. The basic steps are the same as those described in ["Creating an Oracle External Table for Oracle NoSQL Database"](#) on page 6-8.

Creating a Hive External Table for HBase

To provide access to the data in an HBase table, you create a Hive external table over it. Apache provides a storage handler and a SerDe that enable Hive to read the HBase table format.

The following is the basic syntax of a Hive `CREATE TABLE` statement for an external table over an HBase table:

```

CREATE EXTERNAL TABLE tablename colname coltype[, colname coltype,...]
ROW FORMAT
  SERDE 'org.apache.hadoop.hive.hbase.HBaseSerDe'
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
  'serialization.format'='1',
  'hbase.columns.mapping'=':key,value:key,value:

```

See Also:

- *Apache Hive Language Manual DDL* at <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/TruncateTable>
- *Hive HBase Integration* at <https://cwiki.apache.org/confluence/display/Hive/HBaseIntegration#HBaseIntegration-StorageHandlers>
- Class `HBaseSerDe` in the Hive API reference at <http://hive.apache.org/javadocs/r0.13.1/api/hbase-handler/index.html>

Creating the Oracle Database Table for HBase

Use the following syntax to create an external table in Oracle Database that can access the HBase data through a Hive external table:

```
CREATE TABLE tablename(colname colType[, colname colType...])
  ORGANIZATION EXTERNAL
    (TYPE ORACLE_HIVE DEFAULT DIRECTORY DEFAULT_DIR
    ACCESS PARAMETERS
      (access parameters)
    )
  REJECT LIMIT UNLIMITED;
```

In this syntax, you identify the column names and data types. To specify the access parameters, see ["About the SQL CREATE TABLE Statement"](#) on page 6-16.

Creating an Oracle External Table for HDFS Files

The `ORACLE_HDFS` access driver enables you to access many types of data that are stored in HDFS, but which do not have Hive metadata. You can define the record format of text data, or you can specify a SerDe for a particular data format.

You must create the external table for HDFS files manually, and provide all the information the access driver needs to locate the data, and parse the records and fields. The following are some examples of `CREATE TABLE ORGANIZATION EXTERNAL` statements.

Using the Default Access Parameters with `ORACLE_HDFS`

The following statement creates a table named `ORDER` to access the data in all files stored in the `/usr/cust/summary` directory in HDFS:

```
CREATE TABLE ORDER (cust_num VARCHAR2(10),
                    order_num VARCHAR2(20),
                    order_total (NUMBER 8,2))
  ORGANIZATION EXTERNAL (TYPE oracle_hdfs)
  LOCATION ('hdfs:/usr/cust/summary/*');
```

Because no access parameters are set in the statement, the `ORACLE_HDFS` access driver uses the default settings to do the following:

- Connects to the default Hadoop cluster.
- Reads the files as delimited text, and the fields as type `STRING`.

- Assumes that the number of fields in the HDFS files match the number of columns (three in this example).
- Assumes the fields are in the same order as the columns, so that CUST_NUM data is in the first field, ORDER_NUM data is in the second field, and ORDER_TOTAL data is in the third field.
- Rejects any records in which the value causes a data conversion error: If the value for CUST_NUM exceeds 10 characters, the value for ORDER_NUM exceeds 20 characters, or the value of ORDER_TOTAL cannot be converted to NUMBER.

Overriding the Default ORACLE_HDFS Settings

You can use many of the same access parameters with ORACLE_HDFS as ORACLE_HIVE.

Accessing a Delimited Text File

The following example is equivalent to the one shown in ["Overriding the Default ORACLE_HIVE Settings"](#) on page 6-7. The external table access a delimited text file stored in HDFS.

```
CREATE TABLE order (cust_num VARCHAR2(10),
                    order_num VARCHAR2(20),
                    order_date DATE,
                    item_cnt NUMBER,
                    description VARCHAR2(100),
                    order_total (NUMBER8,2)) ORGANIZATION EXTERNAL
    (TYPE oracle_hdfs
    ACCESS PARAMETERS (
        com.oracle.bigdata.colmap:      {"col":"item_cnt", \
                                         "field":"order_line_item_count"}
        com.oracle.bigdata.overflow:    {"action":"TRUNCATE", \
                                         "col":"DESCRIPTION"}
        com.oracle.bigdata.erroropt:    [{"action":"replace", \
                                         "value":"INVALID NUM", \
                                         "col":["CUST_NUM","ORDER_NUM"]} , \
                                         {"action":"reject", \
                                         "col":"ORDER_TOTAL"}]
    )
    LOCATION ("hdfs:/usr/cust/summary/*"));
```

The parameters make the following changes in the way that the ORACLE_HDFS access driver locates the data and handles error conditions:

- com.oracle.bigdata.colmap: Handles differences in column names. ORDER_LINE_ITEM_COUNT in the HDFS files matches the ITEM_CNT column in the external table.
- com.oracle.bigdata.overflow: Truncates string data. Values longer than 100 characters for the DESCRIPTION column are truncated.
- com.oracle.bigdata.erroropt: Replaces bad data. Errors in the data for CUST_NUM or ORDER_NUM set the value to INVALID_NUM.

Accessing Avro Container Files

The next example uses a SerDe to access Avro container files.

```
CREATE TABLE order (cust_num VARCHAR2(10),
                    order_num VARCHAR2(20),
                    order_date DATE,
                    item_cnt NUMBER,
                    description VARCHAR2(100),
```

```
        order_total (NUMBER8,2)) ORGANIZATION EXTERNAL
        (TYPE oracle_hdfs
ACCESS PARAMETERS (
  com.oracle.bigdata.rowformat: \
    SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
  com.oracle.bigdata.fileformat: \
    INPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerInputFormat' \
    OUTPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerOutputFormat'
  com.oracle.bigdata.colmap: { "col": "item_cnt", \
    "field": "order_line_item_count"}
  com.oracle.bigdata.overflow: {"action": "TRUNCATE", \
    "col": "DESCRIPTION"}

LOCATION ("hdfs:/usr/cust/summary/*"));
```

The access parameters provide the following information to the ORACLE_HDFS access driver:

- `com.oracle.bigdata.rowformat`: Identifies the SerDe that the access driver needs to use to parse the records and fields. The files are not in delimited text format.
- `com.oracle.bigdata.fileformat`: Identifies the Java classes that can extract records and output them in the desired format.
- `com.oracle.bigdata.colmap`: Handles differences in column names. ORACLE_HDFS matches `ORDER_LINE_ITEM_COUNT` in the HDFS files with the `ITEM_CNT` column in the external table.
- `com.oracle.bigdata.overflow`: Truncates string data. Values longer than 100 characters for the `DESCRIPTION` column are truncated.

About the SQL CREATE TABLE Statement

The SQL `CREATE TABLE` statement has a clause specifically for creating external tables. The information that you provide in this clause enables the access driver to read data from an external source and prepare the data for the external table.

Basic Syntax

The following is the basic syntax of the `CREATE TABLE` statement for external tables:

```
CREATE TABLE table_name (column_name datatype,
                           column_name datatype[,...])
  ORGANIZATION EXTERNAL (external_table_clause);
```

You specify the column names and data types the same as for any other table. `ORGANIZATION EXTERNAL` identifies the table as an external table.

The *external_table_clause* identifies the access driver and provides the information that it needs to load the data. See ["About the External Table Clause"](#) on page 6-16.

About the External Table Clause

`CREATE TABLE ORGANIZATION EXTERNAL` takes the *external_table_clause* as its argument. It has the following subclauses:

- [TYPE Clause](#)
- [DEFAULT DIRECTORY Clause](#)
- [LOCATION Clause](#)

- [REJECT LIMIT Clause](#)
- [ORACLE_HIVE Access Parameters](#)

See Also: *Oracle Database SQL Language Reference* for the *external_table_clause*

TYPE Clause

The **TYPE** clause identifies the access driver. The type of access driver determines how the other parts of the external table definition are interpreted.

Specify one of the following values for Oracle Big Data SQL:

- **ORACLE_HDFS:** Accesses files in an HDFS directory.
- **ORACLE_HIVE:** Accesses a Hive table.

Note: The **ORACLE_DATAPUMP** and **ORACLE_LOADER** access drivers are not associated with Oracle Big Data SQL.

DEFAULT DIRECTORY Clause

The **DEFAULT DIRECTORY** clause identifies an Oracle Database directory object. The directory object identifies an operating system directory with files that the external table reads and writes.

ORACLE_HDFS and **ORACLE_HIVE** use the default directory solely to write log files on the Oracle Database system.

LOCATION Clause

The **LOCATION** clause identifies the data source.

ORACLE_HDFS LOCATION Clause

The **LOCATION** clause for **ORACLE_HDFS** contains a comma-separated list of file locations. The files must reside in the HDFS file system on the default cluster.

A location can be any of the following:

- A fully qualified HDFS name, such as `/user/hive/warehouse/hive_seed/hive_types`. **ORACLE_HDFS** uses all files in the directory.
- A fully qualified HDFS file name, such as `/user/hive/warehouse/hive_seed/hive_types/hive_types.csv`
- A URL for an HDFS file or a set of files, such as `hdfs:/user/hive/warehouse/hive_seed/hive_types/*`. Just a directory name is invalid.

The file names can contain any pattern-matching character described in [Table 6–2](#).

Table 6–2 *Pattern-Matching Characters*

Character	Description
?	Matches any one character
*	Matches zero or more characters
[abc]	Matches one character in the set {a, b, c}

Table 6–2 (Cont.) Pattern-Matching Characters

Character	Description
[<i>a-b</i>]	Matches one character in the range { <i>a...b</i> }. The character must be less than or equal to <i>b</i> .
[[^] <i>a</i>]	Matches one character that is not in the character set or range { <i>a</i> }. The carat (^) must immediately follow the left bracket, with no spaces.
\ <i>c</i>	Removes any special meaning of <i>c</i> . The backslash is the escape character.
{ <i>ab\,cd</i> }	Matches a string from the set { <i>ab, cd</i> }. The escape character (\) removes the meaning of the comma as a path separator.
{ <i>ab\,c{de\,fh</i> }	Matches a string from the set { <i>ab, cde, cfh</i> }. The escape character (\) removes the meaning of the comma as a path separator.

ORACLE_HIVE LOCATION Clause

Do not specify the `LOCATION` clause for `ORACLE_HIVE`; it raises an error. The data is stored in Hive, and the access parameters and the metadata store provide the necessary information.

REJECT LIMIT Clause

Limits the number of conversion errors permitted during a query of the external table before Oracle Database stops the query and returns an error.

Any processing error that causes a row to be rejected counts against the limit. The reject limit applies individually to each parallel query (PQ) process. It is not the total of all rejected rows for all PQ processes.

ACCESS PARAMETERS Clause

The `ACCESS PARAMETERS` clause provides information that the access driver needs to load the data correctly into the external table. See ["CREATE TABLE ACCESS PARAMETERS Clause"](#) on page 7-5.

About Data Type Conversions

When the access driver loads data into an external table, it verifies that the Hive data can be converted to the data type of the target column. If they are incompatible, then the access driver returns an error. Otherwise, it makes the appropriate data conversion.

Hive typically provides a table abstraction layer over data stored elsewhere, such as in HDFS files. Hive uses a serializer/deserializer (SerDe) to convert the data as needed from its stored format into a Hive data type. The access driver then converts the data from its Hive data type to an Oracle data type. For example, if a Hive table over a text file has a `BIGINT` column, then the SerDe converts the data from text to `BIGINT`. The access driver then converts the data from `BIGINT` (a Hive data type) to `NUMBER` (an Oracle data type).

Performance is better when one data type conversion is performed instead of two. The data types for the fields in the HDFS files should therefore indicate the data that is actually stored on disk. For example, JSON is a clear text format, therefore all data in a JSON file is text. If the Hive type for a field is `DATE`, then the SerDe converts the data from string (in the data file) to a Hive date. Then the access driver converts the data from a Hive date to an Oracle date. However, if the Hive type for the field is string, then the SerDe does not perform a conversion, and the access driver converts the data

from string to an oracle date. Queries against the external table are faster in the second example, because the access driver performs the only data conversion.

[Table 6–3](#) identifies the data type conversions that `ORACLE_HIVE` can make when loading data into an external table.

Table 6–3 Supported Hive to Oracle Data Type Conversions

Hive Data Type	VARCHAR2, CHAR, NCHAR2, NCHAR, CLOB	NUMBER, FLOAT, BINARY NUMBER, BINARY_FLOAT	BLOB	RAW	DATE, TIMESTAMP, TIMESTAMP WITH TZ, TIMESTAMP WITH LOCAL TZ	INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
INT SMALLINT TINYINT BIGINT	yes	yes	yes	yes	no	no
DOUBLE FLOAT	yes	yes	yes	yes	no	no
DECIMAL	yes	yes	no	no	no	no
BOOLEAN	yes ¹	yes ²	yes ²	yes	no	no
BINARY	yes	no	yes	yes	no	no
STRING	yes	yes	yes	yes	yes	yes
TIMESTAMP	yes	no	no	no	yes	no
STRUCT ARRAY UNIONTYPE MAP	yes	no	no	no	no	no

¹ FALSE maps to the string `FALSE`, and TRUE maps to the string `TRUE`.

² FALSE maps to 0, and TRUE maps to 1.

Querying External Tables

Users can query external tables using the SQL `SELECT` statement, the same as they query any other table.

Granting User Access

Users who query the data on a Hadoop cluster must have `READ` access in Oracle Database to the external table and to the database directory object that points to the cluster directory. See ["About the Cluster Directory"](#) on page 6-23.

About Error Handling

By default, a query returns no data if an error occurs while the value of a column is calculated. Processing continues after most errors, particularly those thrown while the column values are calculated.

Use the `com.oracle.bigdata.erroropt` parameter to determine how errors are handled.

About the Log Files

You can use these access parameters to customize the log files:

- [com.oracle.bigdata.log.exec](#)
- [com.oracle.bigdata.log.qc](#)

About Oracle Big Data SQL on Oracle Exadata Database Machine

Oracle Big Data SQL runs exclusively on systems with Oracle Big Data Appliance connected to Oracle Exadata Database Machine. The Oracle Exadata Storage Server Software is deployed on a configurable number of Oracle Big Data Appliance servers. These servers combine the functionality of a CDH node and an Oracle Exadata Storage Server.

The Mammoth utility installs the Big Data SQL software on both Oracle Big Data Appliance and Oracle Exadata Database Machine. The information in this section explains the changes that Mammoth makes to the Oracle Database system.

This section contains the following topics:

- [Starting and Stopping the Big Data SQL Agent](#)
- [About the Common Directory](#)
- [Common Configuration Properties](#)
- [About the Cluster Directory](#)

Note: Oracle SQL Connector for HDFS provides access to Hadoop data for all Oracle Big Data Appliance racks, including those that are not connected to Oracle Exadata Database Machine. However, it does not offer the performance benefits of Oracle Big Data SQL, and it is not included under the Oracle Big Data Appliance license. See *Oracle Big Data Connectors User's Guide*.

Starting and Stopping the Big Data SQL Agent

The `agtctl` utility starts and stops the multithreaded Big Data SQL agent. It has the following syntax:

```
agtctl {startup | shutdown} bds_clustername
```

About the Common Directory

The common directory contains configuration information that is common to all Hadoop clusters. This directory is located on the Oracle Database system under the Oracle home directory. The `oracle` file system user (or whichever user owns the Oracle Database instance) owns the common directory. A database directory named `ORACLE_BIGDATA_CONFIG` points to the common directory.

Common Configuration Properties

The Mammoth installation process creates the following files and stores them in the common directory:

- [bigdata.properties](#)
- [bigdata-log4j.properties](#)

The Oracle DBA can edit these configuration files as necessary.

bigdata.properties

The `bigdata.properties` file in the common directory contains property-value pairs that define the Java class paths and native library paths required for accessing data in HDFS.

These properties must be set:

- `bigdata.cluster.default`
- `java.classpath.hadoop`
- `java.classpath.hive`
- `java.classpath.oracle`

The following list describes all properties permitted in `bigdata.properties`.

bigdata.properties

bigdata.cluster.default

The name of the default Hadoop cluster. The access driver uses this name when the access parameters do not specify a cluster. Required.

Changing the default cluster name might break external tables that were created previously without an explicit cluster name.

bigdata.cluster.list

A comma-separated list of Hadoop cluster names. Optional.

java.classpath.hadoop

The Hadoop class path. Required.

java.classpath.hive

The Hive class path. Required.

java.classpath.oracle

The path to the Oracle JXAD Java JAR file. Required.

java.classpath.user

The path to user JAR files. Optional.

java.libjvm.file

The full file path to the JVM shared library (such as `libjvm.so`). Required.

java.options

A comma-separated list of options to pass to the JVM. Optional.

This example sets the maximum heap size to 2 GB, and verbose logging for Java Native Interface (JNI) calls:

```
Xmx2048m,-verbose=jni
```

LD_LIBRARY_PATH

A colon separated (:) list of directory paths to search for the Hadoop native libraries. Recommended.

If you set this option, then do not set `java.library` path in `java.options`.

[Example 6-1](#) shows a sample `bigdata.properties` file.

Example 6–1 Sample bigdata.properties File

```
# bigdata.properties
#
# Copyright (c) 2014, Oracle and/or its affiliates. All rights reserved.
#
# NAME
#   bigdata.properties - Big Data Properties File
#
# DESCRIPTION
#   Properties file containing parameters for allowing access to Big Data
#   Fixed value properties can be added here
#

java.libjvm.file=$ORACLE_HOME/jdk/jre/lib/amd64/server/libjvm.so
java.classpath.oracle=$ORACLE_HOME/hadoopcore/jlib/*:$ORACLE_
HOME/hadoop/jlib/hver-2/*:$ORACLE_HOME/dbjava/lib/*
java.classpath.hadoop=$HADOOP_HOME/*:$HADOOP_HOME/lib/*
java.classpath.hive=$HIVE_HOME/lib/*
LD_LIBRARY_PATH=$ORACLE_HOME/jdk/jre/lib
bigdata.cluster.default=hadoop_cl_1
```

bigdata-log4j.properties

The bigdata-log4j.properties file in the common directory defines the logging behavior of queries against external tables in the Java code. Any log4j properties are allowed in this file.

[Example 6–2](#) shows a sample bigdata-log4j.properties file with the relevant log4j properties.

Example 6–2 Sample bigdata-log4j.properties File

```
# bigdata-log4j.properties
#
# Copyright (c) 2014, Oracle and/or its affiliates. All rights reserved.
#
# NAME
#   bigdata-log4j.properties - Big Data Logging Properties File
#
# DESCRIPTION
#   Properties file containing logging parameters for Big Data
#   Fixed value properties can be added here
#

bigsql.rootlogger=INFO,console
log4j.rootlogger=DEBUG, file
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{2}:
%m%n
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{2}: %m%n
log4j.logger.oracle.hadoop.sql=ALL, file

bigsql.log.dir=.
bigsql.log.file=bigsql.log
log4j.appender.file.File=$ORACLE_HOME/bigdatalogs/bigdata-log4j.log
```

See Also: Apache Logging Services documentation at
<http://logging.apache.org/log4j/1.2/manual.html>

About the Cluster Directory

The cluster directory contains configuration information for a CDH cluster. Each cluster that Oracle Database will access using Oracle Big Data SQL has a cluster directory. This directory is located on the Oracle Database system under the common directory. For example, a cluster named `bda1_cl_1` would have a directory by the same name (`bda1_cl_1`) in the common directory.

The cluster directory contains the CDH client configuration files for accessing the cluster, such as the following:

- `core-site.xml`
- `hdfs-site.xml`
- `hive-site.xml`
- `mapred-site.xml` (optional)
- `log4j` property files (such as `hive-log4j.properties`)

A database directory object points to the cluster directory. Users who want to access the data in a cluster must have read access to the directory object.

See Also: "[Providing Remote Client Access to CDH](#)" on page 3-2 for a more detailed discussion of Hadoop clients.

About Permissions

The `oracle` operating system user (or whatever user owns the Oracle Database installation directory) must have the following setup:

- READ/WRITE access to the database directory that points to the log directory. These permissions enable the access driver to create the log files, and for the user to read them.
- A corresponding `oracle` operating system user defined on Oracle Big Data Appliance, with READ access in the operating system to the HDFS directory where the source data is stored.

Oracle Big Data SQL Reference

This chapter contains reference information for Oracle Big Data SQL:

- [DBMS_HADOOP PL/SQL Package](#)
- [CREATE TABLE ACCESS PARAMETERS Clause](#)
- [Static Data Dictionary Views for Hive](#)

DBMS_HADOOP PL/SQL Package

The DBMS_HADOOP package contains a function to generate the CREATE EXTERNAL TABLE DDL for a Hive table:

- [CREATE_EXTDDL_FOR_HIVE](#)

CREATE_EXTDDL_FOR_HIVE

This function returns a SQL CREATE TABLE ORGANIZATION EXTERNAL statement for a Hive table. It uses the ORACLE_HIVE access driver.

Syntax

```
DBMS_HADOOP.CREATE_EXTDDL_FOR_HIVE (
  cluster_id      IN    VARCHAR2,
  db_name         IN    VARCHAR2  := NULL,
  hive_table_name IN    VARCHAR2,
  hive_partition  IN    BOOLEAN,
  table_name      IN    VARCHAR2  := NULL,
  perform_ddl     IN    BOOLEAN   DEFAULT FALSE,
  text_of_ddl     OUT   VARCHAR2
);
```

Parameters

Table 7–1 CREATE_EXTDDL_FOR_HIVE Function Parameters

Parameter	Description
cluster_id	Hadoop cluster where the Hive metastore is located
db_name	Name of the Hive database
hive_table_name	Name of the Hive table
hive_partition	Whether the table is partitioned (TRUE) or not (FALSE)
table_name	Name of the Oracle external table to be created. It cannot already exist.
perform_ddl	Whether to execute the generated CREATE TABLE statement (TRUE) or just return the text of the command (FALSE). Do not execute the command automatically if you want to review or modify it.
text_of_ddl	The generated CREATE TABLE ORGANIZATION EXTERNAL statement.

Usage Notes

The Oracle Database system must be configured for Oracle Big Data SQL. See ["About Oracle Big Data SQL on Oracle Exadata Database Machine"](#) on page 6-20.

The data type conversions are based on the default mappings between Hive data types and Oracle data types. See ["About Data Type Conversions"](#) on page 6-18.

Example

The following query returns the CREATE EXTERNAL TABLE DDL for my_hive_table from the default Hive database. The connection to Hive is established using the configuration files in the ORACLE_BIGDATA_CONFIG directory, which identify the location of the HADOOP1 cluster.

```
DECLARE
  DDLtxt VARCHAR2(4000);
BEGIN
  dbms_hadoop.create_extddl_for_hive(
    CLUSTER_ID=>'hadoop1',
```

```
        DB_NAME=>'default',
        HIVE_TABLE_NAME=>'my_hive_table',
        HIVE_PARTITION=>FALSE,
        TABLE_NAME=>'my_xt_oracle',
        PERFORM_DDL=>FALSE,
        TEXT_OF_DDL=>DDLtxt
    );
    dbms_output.put_line(DDLtxt);
END;
/
```

The query returns the text of the following SQL command:

```
CREATE TABLE my_xt_oracle
(
    c0 VARCHAR2(4000),
    c1 VARCHAR2(4000),
    c2 VARCHAR2(4000),
    c3 VARCHAR2(4000)
    ORGANIZATION EXTERNAL
        (TYPE ORACLE_HIVE
            DEFAULT DIRECTORY DEFAULT_DIR
            ACCESS PARAMETERS (
                com.oracle.bigdata.cluster=hadoop1
                com.oracle.bigdata.tablename=default.my_hive_table
            )
        )
    PARALLEL 2 REJECT LIMIT UNLIMITED
```

CREATE TABLE ACCESS PARAMETERS Clause

This section describes the properties that you use when creating an external table that uses the `ORACLE_HDFS` or `ORACLE_HIVE` access drivers. In a `CREATE TABLE ORGANIZATION EXTERNAL` statement, specify the parameters in the `opaque_format_spec` clause of `ACCESS PARAMETERS`.

This section contains the following topics:

- [Syntax Rules for Specifying Properties](#)
- [ORACLE_HDFS Access Parameters](#)
- [ORACLE_HIVE Access Parameters](#)
- Alphabetical list of properties

Syntax Rules for Specifying Properties

The properties are set using keyword-value pairs in the SQL `CREATE TABLE ACCESS PARAMETERS` clause and in the configuration files. The syntax must obey these rules:

- The format of each keyword-value pair is a *keyword*, a colon or equal sign, and a *value*. The following are valid keyword-value pairs:

```
keyword=value
keyword:value
```

The value is everything from the first non-whitespace character after the separator to the end of the line. Whitespace between the separator and the value is ignored. Trailing whitespace for the value is retained.

- A property definition can be on one line or multiple lines.
- A line terminator is a line feed, a carriage return, or a carriage return followed by line feeds.
- When a property definition spans multiple lines, then precede the line terminators with a backslash (escape character), except on the last line. In this example, the value of the `Keyword1` property is `Value part 1 Value part 2 Value part 3`.

```
Keyword1= Value part 1 \
          Value part 2 \
          Value part 3
```

- You can create a *logical line* by stripping each physical line of leading whitespace and concatenating the lines. The parser extracts the property names and values from the logical line.
- You can embed special characters in a property name or property value by preceding a character with a backslash (escape character), indicating the substitution. [Table 7–2](#) describes the special characters.

Table 7–2 Special Characters in Properties

Escape Sequence	Character
<code>\b</code>	Backspace (<code>\u0008</code>)
<code>\t</code>	Horizontal tab (<code>\u0009</code>)
<code>\n</code>	Line feed (<code>\u000a</code>)
<code>\f</code>	Form feed (<code>\u000c</code>)
<code>\r</code>	Carriage return (<code>\u000d</code>)
<code>\"</code>	Double quote (<code>\u0022</code>)
<code>\'</code>	Single quote (<code>\u0027</code>)
<code>\\</code>	Backslash (<code>\u005c</code>) When multiple backslashes are at the end of the line, the parser continues the value to the next line only for an odd number of backslashes.
<code>\uxxxx</code>	2-byte, big-endian, Unicode code point. When a character requires two code points (4 bytes), the parser expects <code>\u</code> for the second code point.

ORACLE_HDFS Access Parameters

The access parameters for the ORACLE_HDFS access driver provide the metadata needed to locate the data in HDFS and generate a Hive table over it.

Default Parameter Settings for ORACLE_HDFS

If you omit all access parameters from the CREATE TABLE statement, then ORACLE_HDFS uses the following default values:

```
com.oracle.bigdata.rowformat=DELIMITED
com.oracle.bigdata.fileformat=TEXTFILE
com.oracle.bigdata.overflow={"action":"truncate"}
com.oracle.bigdata.erroropt={"action":"setnull"}
```

Optional Parameter Settings for ORACLE_HDFS

ORACLE_HDFS supports the following optional com.oracle.bigdata parameters, which you can specify in the opaque_format_spec clause:

- `com.oracle.bigdata.colmap`
- `com.oracle.bigdata.erroropt`
- `com.oracle.bigdata.fields`
- `com.oracle.bigdata.fileformat`
- `com.oracle.bigdata.log.exec`
- `com.oracle.bigdata.log.qc`
- `com.oracle.bigdata.overflow`
- `com.oracle.bigdata.rowformat`

Example 7–1 shows a CREATE TABLE statement in which multiple access parameters are set.

Example 7–1 Setting Multiple Access Parameters for ORACLE_HDFS

```
CREATE TABLE ORDER (CUST_NUM VARCHAR2(10),
                     ORDER_NUM VARCHAR2(20),
                     ORDER_DATE DATE,
                     ITEM_CNT NUMBER,
                     DESCRIPTION VARCHAR2(100),
                     ORDER_TOTAL (NUMBER8,2)) ORGANIZATION EXTERNAL
    (TYPE ORACLE_HDFS
    ACCESS PARAMETERS (
        com.oracle.bigdata.fields: (CUST_NUM,          \
                                   ORDER_NUM,          \
                                   ORDER_DATE,         \
                                   ORDER_LINE_ITEM_COUNT, \
                                   DESCRIPTION,        \
                                   ORDER_TOTAL)
        com.oracle.bigdata.colMap:    {"col":"item_cnt", \
                                       "field":"order_line_item_count"}
        com.oracle.bigdata.overflow: {"action":"TRUNCATE", \
                                       "col":"DESCRIPTION"}
        com.oracle.bigdata.errorOpt: [{"action":"replace", \
                                       "value":"INVALID NUM", \
                                       "col":["CUST_NUM","ORDER_NUM"]} , \
                                       {"action":"reject", \
```

```
                                "col": "ORDER_TOTAL"}]
    )
LOCATION ("hdfs:/usr/cust/summary/*");
```


ORACLE_HIVE Access Parameters

ORACLE_HIVE retrieves metadata about external data sources from the Hive catalog. The default mapping of Hive data to columns in the external table are usually appropriate. However, some circumstances require special parameter settings, or you might want to override the default values for reasons of your own.

Default Parameter Settings for ORACLE_HIVE

If you omit all access parameters from the CREATE TABLE statement, then ORACLE_HIVE uses the following default values:

```
com.oracle.bigdata.tablename=name of external table
com.oracle.bigdata.overflow={"action":"truncate"}
com.oracle.bigdata.erroropt={"action":"setnull"}
```

Optional Parameter Values for ORACLE_HIVE

ORACLE_HIVE supports the following optional com.oracle.bigdata parameters, which you can specify in the opaque_format_spec clause:

- `com.oracle.bigdata.colmap`
- `com.oracle.bigdata.erroropt`
- `com.oracle.bigdata.log.exec`
- `com.oracle.bigdata.log.qc`
- `com.oracle.bigdata.overflow`
- `com.oracle.bigdata.tablename`

Example 7–2 shows a CREATE TABLE statement in which multiple access parameters are set.

Example 7–2 Setting Multiple Access Parameters for ORACLE_HIVE

```
CREATE TABLE ORDER (cust_num VARCHAR2(10),
                     order_num VARCHAR2(20),
                     order_date DATE,
                     item_cnt NUMBER,
                     description VARCHAR2(100),
                     order_total (NUMBER8,2)) ORGANIZATION EXTERNAL
(TYPE oracle_hive
 ACCESS PARAMETERS (
   com.oracle.bigdata.tableName: order_db.order_summary
   com.oracle.bigdata.colMap:    {"col":"ITEM_CNT", \
                                "field":"order_line_item_count"}
   com.oracle.bigdata.overflow: {"action":"ERROR", \
                                "col":"DESCRIPTION"}
   com.oracle.bigdata.errorOpt: [{"action":"replace", \
                                "value":"INV_NUM" , \
                                "col":["CUST_NUM","ORDER_NUM"]} ,\
                                {"action":"reject", \
                                "col":"ORDER_TOTAL"}]
 ));
```

com.oracle.bigdata.colmap

Maps a column in the source data to a column in the Oracle external table. Use this property when the source field names exceed the maximum length of Oracle column names, or when you want to use different column names in the external table.

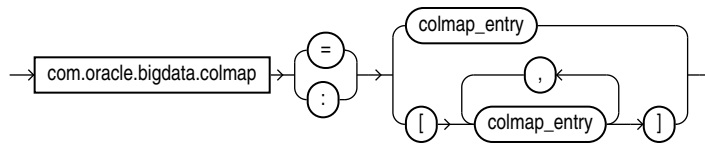
Default Value

A column in the external table with the same name as the Hive column

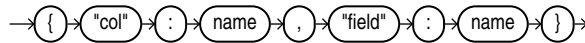
Syntax

A JSON document with the keyword-value pairs is shown in the following diagram:

colmap ::=



colmap_entry ::=



Semantics

"col":name

"col": The keyword must be lowercase and enclosed in quotation marks.

name: The name of a column in the Oracle external table. It is case sensitive and must be enclosed in quotation marks.

"field":name

"field": The keyword must be lowercase and enclosed in quotation marks.

name: The name of a field in the data source. It is not case sensitive, but it must be enclosed in quotation marks. See ["Syntax Rules for Specifying Properties"](#) on page 7-6.

Example

This example maps a Hive column named ORDER_LINE_ITEM_COUNT to an Oracle column named ITEM_CNT:

```
com.oracle.bigdata.colMap={"col":"ITEM_CNT", \
                           "field":"order_line_item_count"}
```

com.oracle.bigdata.datamode

Specifies the method that SmartScan uses to scan a Hadoop data source. The method can make a significant difference in performance.

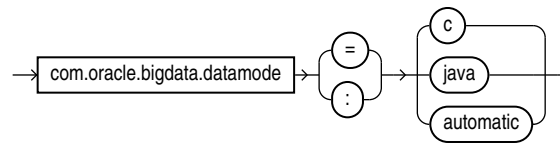
Default Value

java

Syntax

A JSON document with the keyword-value pairs shown in the following diagram:

datamode ::=



Semantics

automatic

Automatically selects the appropriate mode, based on the metadata. It selects `c` mode if possible, or `java` mode if the data contains formats that are not supported by `c` mode.

c

Uses Java to read the file buffers, but C code to process the data and convert it to Oracle format. Specify this mode for delimited data.

If the data contains formats that the C code does not support, then it returns an error.

java

Uses the Java SerDes and InputFormats to process the data and convert it to Oracle format. Specify this mode for Parquet, RCFile, and other data formats that require a SerDe.

com.oracle.bigdata.erroropt

Describes how to handle errors that occur while the value of a column is calculated.

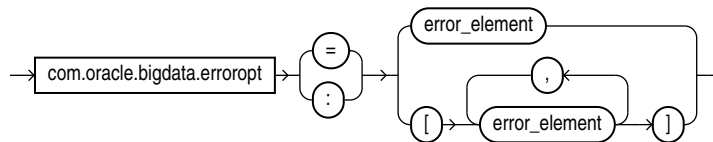
Default Value

```
{"action": "setnull"}
```

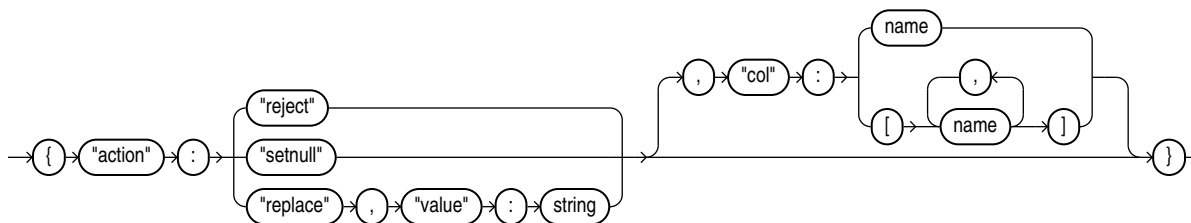
Syntax

A JSON document with the keyword-value pairs is shown in the following diagram:

erroropt ::=



error_element ::=



Semantics

The "action", "reject", "setnull", "replace", "value", and "col" keywords must be lowercase and enclosed in quotation marks. See ["Syntax Rules for Specifying Properties"](#) on page 7-6.

"action":value

value: One of these keywords:

- "reject": Does not load any rows.
- "setnull": Sets the column to NULL.
- "replace": Sets the column to the specified value.

"value":string

string: Replaces a bad value in the external table. It must be enclosed in quotation marks.

"col":name

name: Identifies a column in an external table. The column name is case sensitive, must be enclosed in quotation marks, and can be listed only once.

Example

This example sets the value of the CUST_NUM or ORDER_NUM columns to INVALID if the Hive value causes an error. For any other columns, an error just causes the Hive value to be rejected.

```
com.oracle.bigdata.errorOpt: {"action": "replace", \
                              "value": "INVALID", \
                              "col": ["CUST_NUM", "ORDER_NUM"]}
```

com.oracle.bigdata.fields

Lists the field names and data types of the data source.

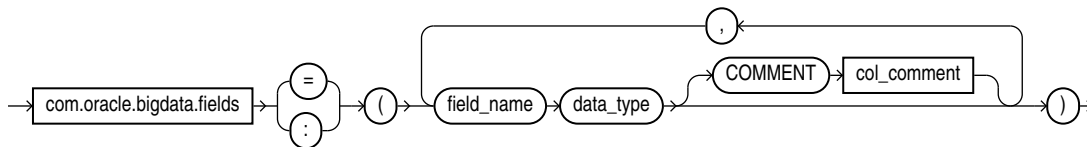
Default Value

Not defined

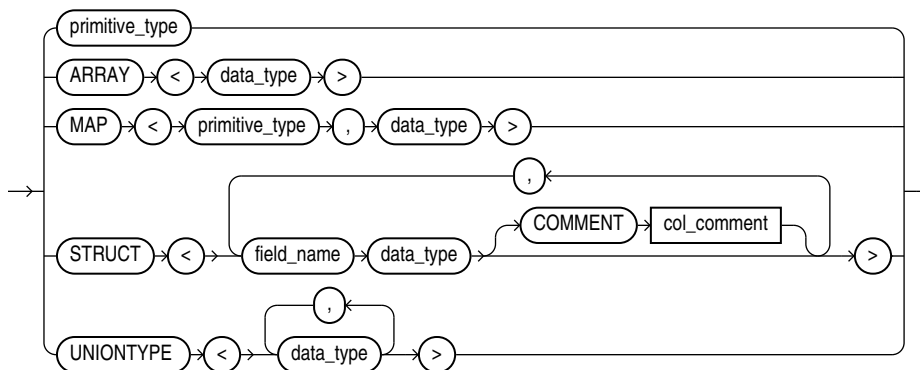
Syntax

A JSON document with the keyword-value pairs is shown in the following diagram:

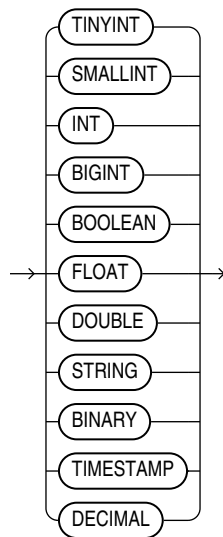
fields ::=



data_type ::=



primitive_type ::=



Semantics

The syntax is the same as a field list for a Hive table. If you split the field list across multiple lines, you must use a backslash to escape the new line characters.

field_name

The name of the Hive field. Use only alphanumeric characters and underscores (_). The maximum length is 128 characters. Field names are case-insensitive.

data_type

The data type of the Hive field. Optional; the default is `STRING`. The character set must be UTF8.

The data type can be complex or primitive:

Hive Complex Data Types

- `ARRAY`: Indexable list
- `MAP`: Key-value tuples
- `STRUCT`: List of elements
- `UNIONTYPE`: Multiple data types

Hive Primitive Data Types

- `INT`: 4 byte integer
- `BIGINT`: 8 byte integer
- `SMALLINT`: 2 byte integer
- `TINYINT`: 1 byte integer
- `BOOLEAN`: `TRUE` or `FALSE`
- `FLOAT`: single precision
- `DOUBLE`: double precision
- `STRING`: character sequence

See Also: "Data Types" in the *Apache Hive Language Manual* at

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types>

COMMENT *col_comment*

A string literal enclosed in single quotation marks, which is stored as metadata for the Hive table (`comment` property of `TBLPROPERTIES`).

com.oracle.bigdata.fileformat

Describes the row format of the data source, based on the `ROW FORMAT` clause for a Hive table generated by `ORACLE_HDFS`.

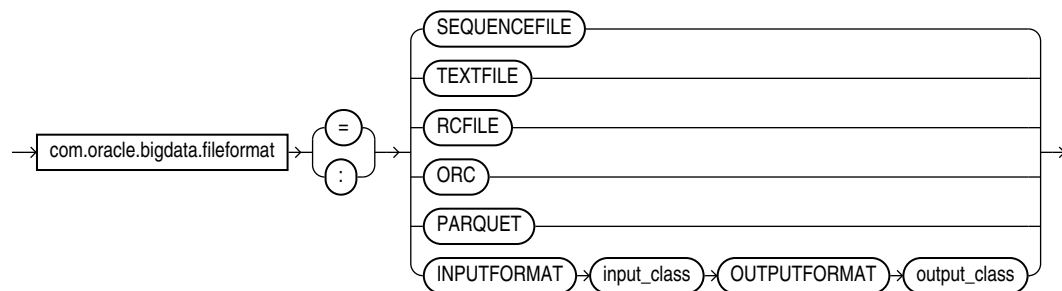
Default Value

TEXTFILE

Syntax

A JSON document with the keyword-value pairs is shown in the following diagram.

fileformat ::=



Semantics

ORC

Optimized row columnar file format

PARQUET

Column-oriented, binary file format

RCFILE

Record columnar file format

SEQUENCEFILE

Compressed file format

TEXTFILE

Plain text file format

INPUTFORMAT

Identifies a Java class that can extract records from the data file.

OUTPUTFORMAT

Identifies a Java class that can format the output records in the desired format.

com.oracle.bigdata.log.exec

Specifies how the access driver generates log files generated by the C code for a query, when it is running as parallel processes on CDH.

The access driver does not create or write log files when executing on a Hadoop cluster node; the parallel query processes write them. The log files from the Java code are controlled by `log4j` properties, which are specified in the configuration file or the access parameters. See ["bigdata-log4j.properties"](#) on page 6-22.

Default Value

Not defined (no logging)

Syntax

```
[directory_object:]file_name_template
```

Semantics

directory_object

The Oracle directory object for the HDFS path on the Hadoop cluster where the log file is created.

file_name_template

A string used to generate file names. [Table 7–2](#) describes the optional variables that you can use in the template.

Table 7–3 Variables for com.oracle.bigdata.log.exec

Variable	Value
%p	Operating system process identifier (PID)
%a	A number that uniquely identifies the process.
%%	A percent sign (%)

Example

The following example generates log file names that include the PID and a unique number, such as `xtlogp_hive14_3413_57`:

```
com.oracle.bigdata.log.exec= xtlogp_hive14_%p_%a
```

com.oracle.bigdata.log.qc

Specifies how the access driver generates log files for a query.

Default Value

Not defined (no logging)

Syntax

[directory_object:]file_name_template

Semantics

directory_object

Name of an Oracle directory object that points to the path where the log files are written. If this value is omitted, then the logs are written to the default directory for the external table.

file_name_template

A string used to generate file names. [Table 7–4](#) describes the optional variables that you can use in the string.

Table 7–4 Variables for com.oracle.bigdata.log.qc

Variable	Value
%p	Operating system process identifier (PID)
%%	A percent sign (%)

Example

This example creates log file names that include the PID and a percent sign, such as xtlogp_hive213459_%:

```
com.oracle.bigdata.log.qc= xtlogp_hive21%p_%%
```

com.oracle.bigdata.overflow

Describes how to handle string data that is too long for the columns in the external table. The data source can be character or binary. For Hive, the data source can also be STRUCT, UNIONTYPES, MAP, or ARRAY.

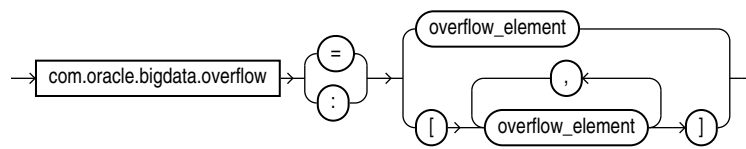
Default Value

```
{"action": "truncate"}
```

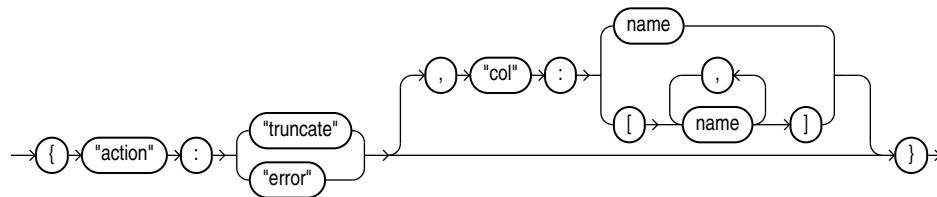
Syntax

A JSON document with the keyword-value pairs is shown in the following diagram:

overflow ::=



overflow_element ::=



Semantics

The "action", "truncate", "error", and "col" tags must be lowercase and enclosed in quotation marks. See ["Syntax Rules for Specifying Properties"](#) on page 7-6.

"action":value

The value of "action" can be one of the following keywords:

- truncate: Shortens the data to fit the column.
- error: Throws an error. The [com.oracle.bigdata.erroropt](#) property controls the result of the error.

"col":name

name: Identifies a column in the external table. The name is case sensitive and must be enclosed in quotation marks.

Example

This example truncates the source data for the DESCRIPTION column, if it exceeds the column width:

```
com.oracle.bigdata.overflow={"action": "truncate", \
                             "col": "DESCRIPTION"}
```

com.oracle.bigdata.rowformat

Provides the information the access driver needs to extract fields from the records in a file.

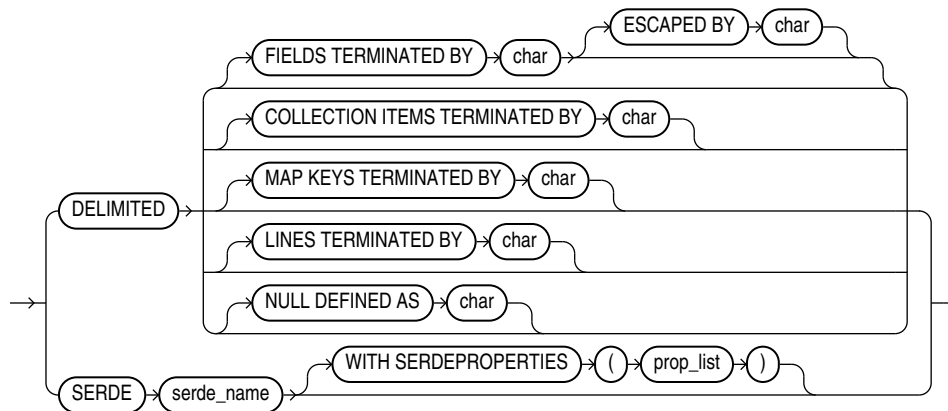
Default Value

DELIMITED

Syntax

A JSON document with the keyword-value pairs is shown in the following diagram.

rowformat ::=



Semantics

DELIMITED

Describes the characters used to delimit the fields in a record:

- **FIELDS TERMINATED BY:** The character that delimits every field in the record. The optional **ESCAPED BY** character precedes the delimit character when it appears within a field value.
- **COLLECTION ITEMS TERMINATED BY:** The character that marks the end of an array element.
- **MAP KEYS TERMINATED BY:** The character that marks the end of an entry in a MAP field.
- **LINES TERMINATED BY:** The character that marks the end of a record.
- **NULL DEFINED AS:** The character that indicates a null value.

SERDE

Identifies a SerDe that can parse the data and any properties of the SerDe that the access driver might need.

Example

This example specifies a SerDe for an Avro container file:

```
com.oracle.bigdata.rowformat:
```

```
SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
```

The next example specifies a SerDe for a file containing regular expressions:

```
com.oracle.bigdata.rowformat=\
SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe' \
WITH SERDEPROPERTIES \
  ("input.regex" = "(\\\\d{6}) (\\\\d{5}) (.{29}) .*")
```

com.oracle.bigdata.tablename

Identifies the Hive table that contains the source data.

Default Value

DEFAULT.*external_table_name*

Syntax

[hive_database_name.]table_name

Semantics

The maximum length of *hive_database_name* and *table_name* is 128 UTF-8 characters (512 bytes).

hive_database_name

The Hive database where the source data resides. DEFAULT is the name of the initial Hive database.

table_name

The Hive table with the data. If you omit *table_name*, then ORACLE_HIVE searches for a Hive table with the same name as the external table. Table names are case-insensitive.

Example

This setting indicates that the source data is in a table named ORDER_SUMMARY in the Hive ORDER_DB database:

```
com.oracle.bigdata.tablename ORDER_DB.ORDER_SUMMARY
```

Static Data Dictionary Views for Hive

The Oracle Database catalog contains several static data dictionary views for Hive tables. You can query these data dictionary views to discover information about the Hive tables that you can access.

For you to access any Hive databases from Oracle Database, you must have read privileges on the `ORACLE_BIGDATA_CONFIG` directory object.

- [ALL_HIVE_DATABASES](#)
- [ALL_HIVE_TABLES](#)
- [ALL_HIVE_COLUMNS](#)
- [DBA_HIVE_DATABASES](#)
- [DBA_HIVE_TABLES](#)
- [DBA_HIVE_COLUMNS](#)
- [USER_HIVE_DATABASES](#)
- [USER_HIVE_TABLES](#)
- [USER_HIVE_COLUMNS](#)

ALL_HIVE_DATABASES

ALL_HIVE_DATABASES describes all databases in the Hive metastore accessible to the current user.

Related Views

- DBA_HIVE_DATABASES describes all the databases in the Hive metastore.
- USER_HIVE_DATABASES describes the databases in the Hive metastore owned by the current user.

Column	Datatype	NULL	Description
CLUSTER_ID	VARCHAR2 (4000)	NOT NULL	Hadoop cluster where the Hive metastore is located
DATABASE_NAME	VARCHAR2 (4000)	NOT NULL	Hive database name
DESCRIPTION	VARCHAR2 (4000)		Hive database description
DB_LOCATION	VARCHAR2 (4000)	NOT NULL	
HIVE_URI	VARCHAR2 (4000)		Hive database URI

See Also:

- ["DBA_HIVE_DATABASES"](#) on page 7-27
- ["USER_HIVE_DATABASES"](#) on page 7-30

ALL_HIVE_TABLES

ALL_HIVE_TABLES describes all tables in the Hive metastore accessible to the current user.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See ["About the Common Directory"](#) on page 6-20.

Related Views

- DBA_HIVE_TABLES describes all tables in the Hive metastore.
- USER_HIVE_TABLES describes the tables in the database owned by the current user in the Hive metastore.

Column	Datatype	NULL	Description
CLUSTER_ID	VARCHAR2 (4000)	NOT NULL	Hadoop cluster where the Hive metastore is located
DATABASE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive database
TABLE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive table
LOCATION	VARCHAR2 (4000)		
NO_OF_COLS	NUMBER		Number of columns in the Hive table
CREATION_TIME	DATE		Time when the table was created
LAST_ACCESSED_TIME	DATE		Time of most recent access
OWNER	VARCHAR2 (4000)		Owner of the Hive table
TABLE_TYPE	VARCHAR2 (4000)	NOT NULL	Type of Hive table, such as external or managed
PARTITIONED	VARCHAR2 (4000)		Whether the table is partitioned (YES) or not (NO)
NO_OF_PART_KEYS	NUMBER		Number of partitions
INPUT_FORMAT	VARCHAR2 (4000)		Input format
OUTPUT_FORMAT	VARCHAR2 (4000)		Output format
SERIALIZATION	VARCHAR2 (4000)		SerDe serialization information
COMPRESSED	NUMBER		Whether the table is compressed (YES) or not (NO)
HIVE_URI	VARCHAR2 (4000)		Hive database URI

See Also:

- ["DBA_HIVE_TABLES"](#) on page 7-28
- ["USER_HIVE_TABLES"](#) on page 7-31

ALL_HIVE_COLUMNS

ALL_HIVE_COLUMNS describes the columns of all Hive tables accessible to the current user.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See ["About the Common Directory"](#) on page 6-20.

Related Views

- DBA_HIVE_COLUMNS describes the columns of all tables in the Hive metastore.
- USER_HIVE_COLUMNS describes the columns of the tables in the Hive database owned by the current user.

Column	Datatype	NULL	Description
CLUSTER_ID	VARCHAR2 (4000)	NOT NULL	Hadoop cluster where the Hive metastore is located
DATABASE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive database; if blank, then the default database
TABLE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive table
COLUMN_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive column
HIVE_COLUMN_TYPE	VARCHAR2 (4000)	NOT NULL	Data type of the Hive column
ORACLE_COLUMN_TYPE	VARCHAR2 (4000)	NOT NULL	Oracle data type equivalent to Hive data type
LOCATION	VARCHAR2 (4000)		
OWNER	VARCHAR2 (4000)		Owner of the Hive table
CREATION_TIME	DATE		Time when the table was created
HIVE_URI	VARCHAR2 (4000)		Hive database URI

See Also:

- ["DBA_HIVE_COLUMNS"](#) on page 7-29
- ["USER_HIVE_COLUMNS"](#) on page 7-32

DBA_HIVE_DATABASES

DBA_HIVE_DATABASES describes all the databases in the Hive metastore. Its columns are the same as those in ALL_HIVE_DATABASES.

See Also: ["ALL_HIVE_DATABASES"](#) on page 7-24

DBA_HIVE_TABLES

DBA_HIVE_TABLES describes all tables in the Hive metastore. Its columns are the same as those in ALL_HIVE_TABLES.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. See ["About the Common Directory"](#) on page 6-20.

See Also: ["ALL_HIVE_TABLES"](#) on page 7-25

DBA_HIVE_COLUMNS

DBA_HIVE_COLUMNS describes the columns of all tables in the Hive metastore. Its columns are the same as those in ALL_HIVE_COLUMNS.

See Also: ["ALL_HIVE_COLUMNS"](#) on page 7-26

USER_HIVE_DATABASES

USER_HIVE_DATABASES describes the databases in the Hive metastore owned by the current user. Its columns (except for OWNER) are the same as those in ALL_HIVE_DATABASES.

See Also: ["ALL_HIVE_DATABASES"](#) on page 7-24

USER_HIVE_TABLES

USER_HIVE_TABLES describes the tables in the database owned by the current user in the Hive metastore. Its columns (except for OWNER) are the same as those in ALL_HIVE_TABLES.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See ["About the Common Directory"](#) on page 6-20.

See Also: ["ALL_HIVE_TABLES"](#) on page 7-25

USER_HIVE_COLUMNS

USER_HIVE_COLUMNS describes the columns of the tables in the Hive database owned by the current user. Its columns (except for OWNER) are the same as those in ALL_HIVE_COLUMNS.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See ["About the Common Directory"](#) on page 6-20.

See Also: ["ALL_HIVE_COLUMNS"](#) on page 7-26

Copying Oracle Tables to Hadoop

This chapter describes how to use Copy to BDA to copy tables in an Oracle database to Hadoop. It contains the following sections:

- [What Is Copy to BDA?](#)
- [Getting Started Using Copy to BDA](#)
- [Installing Copy to BDA](#)
- [Generating the Data Pump Files](#)
- [Creating a Hive Table](#)
- [Example Using the Sample Schemas](#)

What Is Copy to BDA?

Copy to BDA enables you to copy tables from an Oracle database into Hadoop. After generating Data Pump format files from the tables and copying the files to HDFS, you can use Apache Hive to query the data. Hive can process the data locally without accessing Oracle Database. When the Oracle table changes, you can refresh the copy in Hadoop. Copy to BDA is primarily useful for Oracle tables that are relatively static, and thus do not require frequent refreshes.

Copy to BDA is licensed under Oracle Big Data SQL. You must have an Oracle Big Data SQL license to use Copy to BDA.

Getting Started Using Copy to BDA

Take the following steps to use Copy to BDA:

1. Ensure that your system meets the prerequisites, and that the required software is installed on Oracle Big Data Appliance and Oracle Exadata Database Machine.
See ["Installing Copy to BDA"](#) on page 8-2.
2. On Oracle Exadata Database Machine, connect to Oracle Database and generate Data Pump format files containing the table data and metadata.
See ["Generating the Data Pump Files"](#) on page 8-2.
3. Copy the files to HDFS on Oracle Big Data Appliance.
See ["Copying the Files to HDFS"](#) on page 8-4.
4. Connect to Apache Hive and create an external table from the files.
See ["Creating a Hive Table"](#) on page 8-4.

5. Query this Hive table the same as you would any other Hive table.

Installing Copy to BDA

Copy to BDA is available only on Oracle Exadata Database Machine connected to Oracle Big Data Appliance.

Prerequisites for Copy to BDA

Oracle Exadata Database Machine must comply with the following requirements:

- Configured on the same InfiniBand or client network as Oracle Big Data Appliance. Oracle recommends an InfiniBand connection between Oracle Exadata Database Machine and Oracle Big Data Appliance, but it is not required.
- Runs Oracle Database 11.2 or later.

Copy to BDA supports earlier releases than Oracle Big Data SQL.

Installing Copy to BDA on Oracle Big Data Appliance

Copy to BDA is a component of Oracle Big Data SQL, which is an installation option on Oracle Big Data Appliance. You can enable Oracle Big Data SQL either during the initial software installation or at a later time using the standard methods for enabling and disabling services. See ["Performing the Installation"](#) on page 6-3.

Installing Copy to BDA on Oracle Exadata Database Machine

Copy to BDA only requires a Hadoop client on Oracle Exadata Database Machine. It does not employ the additional software required by Oracle Big Data SQL.

If you plan to use Oracle Big Data SQL also, then the Hadoop client is created automatically when you run the `bds-exa-install.sh` installation script. You do not need to take any additional steps. See ["Running the Post-Installation Script for Oracle Big Data SQL"](#) on page 6-4.

If you do not plan to use Oracle Big Data SQL at this time, then you can install the Hadoop client manually instead of running the script. For example, Oracle Big Data SQL might not be supported with the version of Oracle Database you are using. Or you might want to avoid installing a database patch and other software that you do not currently need. See ["Providing Remote Client Access to CDH"](#) on page 3-2.

Generating the Data Pump Files

The SQL `CREATE TABLE` statement has a clause specifically for creating external tables, in which you specify the `ORACLE_DATAPUMP` access driver. The information that you provide in this clause enables the access driver to generate a Data Pump format file that contains the data and metadata from the Oracle database table.

This section contains the following topics:

- [About Data Pump Format Files](#)
- [Identifying the Target Directory](#)
- [About the CREATE TABLE Syntax](#)
- [Copying the Files to HDFS](#)

About Data Pump Format Files

Data Pump files are typically used to move data and metadata from one database to another. Copy to BDA uses this file format to copy data from an Oracle database to HDFS.

To generate Data Pump format files, you create an external table from an existing Oracle table. An **external table** in Oracle Database is an object that identifies and describes the location of data outside of a database. External tables use **access drivers** to parse and format the data. For Copy to BDA, you use the `ORACLE_DATAPUMP` access driver. It copies the data and metadata from internal Oracle tables and populates the Data Pump format files of the external table.

Identifying the Target Directory

You must have read and write access to a database directory in Oracle Database. Only Oracle Database users with the `CREATE ANY DIRECTORY` system privilege can create directories.

This example creates a database directory named `EXPORTDIR` that points to the `/exportdir` directory on Oracle Exadata Database Machine:

```
SQL> CREATE DIRECTORY exportdir AS '/exportdir';
```

About the CREATE TABLE Syntax

The following is the basic syntax of the `CREATE TABLE` statement for Data Pump format files:

```
CREATE TABLE table_name
  ORGANIZATION EXTERNAL (
    TYPE oracle_datapump
    DEFAULT DIRECTORY database_directory
    LOCATION (' filename1.dmp', ' filename2.dmp' ...)
  ) PARALLEL n
AS SELECT * FROM tablename;
```

DEFAULT DIRECTORY

Identifies the database directory that you created for this purpose. See ["Identifying the Target Directory"](#) on page 8-3.

LOCATION

Lists the names of the Data Pump files to be created. The number of names should match the degree of parallelism (DOP) specified by the `PARALLEL` clause. Otherwise, the DOP drops to the number of files.

The number of files and the degree of parallelism affect the performance of Oracle Database when generating the Data Pump format files. They do not affect querying performance in Hive.

PARALLEL

Sets the degree of parallelism (DOP). Use the maximum number that your Oracle DBA permits you to use. By default the DOP is 1, which is serial processing. Larger numbers enable parallel processing.

AS SELECT

Use the full SQL `SELECT` syntax for this clause. It is not restricted. The *tablename* identifies the Oracle table to be copied to HDFS.

See Also: For descriptions of these parameters:

- *Oracle Database SQL Language Reference*
- *Oracle Database Utilities*

Copying the Files to HDFS

The Oracle Big Data SQL installation installs Hadoop client files to Oracle Exadata Database Machine. The Hadoop client installation enables you to use Hadoop commands to copy the Data Pump files to HDFS. You must have write privileges on the HDFS directory.

To copy the `dmp` files into HDFS, use the `hadoop fs -put` command. This example copies the files into the HDFS `customers` directory owned by the `oracle` user:

```
$ hadoop fs -put customers*.dmp /user/oracle/customers
```

Creating a Hive Table

To provide access to the data in the Data Pump files, you create a Hive external table over the Data Pump files. Copy to BDA provides SerDes that enable Hive to read the files. These SerDes are read only, so you cannot use them to write to the files.

See Also: *Apache Hive Language Manual DDL* at

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/TruncateTable>

About Hive External Tables

For external tables, Hive loads the table metadata into its metastore. The data remains in its original location, which you identify in the `LOCATION` clause. If you drop an external table using a HiveQL `DROP TABLE` statement, then only the metadata is discarded, while the external data remains unchanged. In this respect, Hive handles external tables in fundamentally the same way as Oracle Database.

External tables support data sources that are shared by multiple programs. In this case, you use Oracle Database to update the data and then generate a new file. You can overwrite the old HDFS files with the updated files while leaving the Hive metadata intact.

The following is the basic syntax of a Hive `CREATE TABLE` statement for creating a Hive external table for use with a Data Pump format file:

```
CREATE EXTERNAL TABLE tablename
ROW FORMAT
  SERDE 'oracle.hadoop.hive.datapump.DPSerDe'
STORED AS
  INPUTFORMAT 'oracle.hadoop.hive.datapump.DPInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 'hdfs_directory'
```

About Column Mappings

The Hive table columns automatically have the same names as the Oracle columns, which are provided by the metadata stored in the Data Pump files. In this release, any user-specified column definitions are ignored.

About Data Type Conversions

Copy to BDA automatically converts the data in an Oracle table to an appropriate Hive data type. [Table 8–1](#) shows the default mappings between Oracle and Hive data types.

Table 8–1 Oracle to Hive Data Type Conversions

Oracle Data Type	Hive Data Type
NUMBER	INT when the scale is 0 and the precision is less than 10 BIGINT when the scale is 0 and the precision is less than 19 DECIMAL when the scale is greater than 0 or the precision is greater than 19
BINARY_DOUBLE	DOUBLE
BINARY_FLOAT	FLOAT
CHAR	CHAR
NCHAR	
VARCHAR2	VARCHAR
NVARCHAR2	
DATE	TIMESTAMP
TIMESTAMP	TIMESTAMP
TIMESTAMPtz ¹	Unsupported
TIMESTAMPPLTZ	
RAW	BINARY

¹ To copy `TIMESTAMPtz` and `TIMESTAMPPLTZ` data to Hive, cast the columns to `TIMESTAMP` when exporting them to the Data Pump files. Hive does not have a data type that supports time zones or time offsets.

Example Using the Sample Schemas

This example shows all steps in the process of creating a Hive table from an Oracle table using Copy to BDA.

About the Sample Data

The Oracle tables are from the Sales History (SH) sample schema. The `CUSTOMERS` table provides extensive information about individual customers, including names, addresses, telephone numbers, birth dates, and credit limits. The `COUNTRIES` table provides a list of countries, and identifies regions and subregions.

This query shows a small selection of data in the `CUSTOMERS` table:

```
SELECT cust_first_name first_name,
       cust_last_name last_name,
       cust_gender gender,
       cust_year_of_birth birth
FROM customers
ORDER BY cust_city, last_name
FETCH FIRST 10 ROWS ONLY;
```

The query returns the following rows:

```
FIRST_NAME    LAST_NAME    GENDER    BIRTH
-----
```

Lise	Abbey	F	1963
Lotus	Alden	M	1958
Emmanuel	Aubrey	M	1933
Phil	Ball	M	1956
Valentina	Bardwell	F	1965
Lolita	Barkley	F	1966
Heloise	Barnes	M	1980
Royden	Barrett	M	1937
Gilbert	Braun	M	1984
Portia	Capp	F	1948

To reproduce this example, install the sample schemas in Oracle Database and connect as the SH user.

See Also: *Oracle Database Sample Schemas* for descriptions of the tables and installation instructions for the schemas.

Creating the EXPDIR Database Directory

These SQL statements create a local database directory named EXPDIR and grant access to the SH user:

```
SQL> CREATE DIRECTORY expdir AS '/expdir';
```

Directory created.

```
SQL> GRANT READ, WRITE ON DIRECTORY expdir TO SH;
```

Grant succeeded.

Creating Data Pump Format Files for Customer Data

The following examples show how to create the Data Pump files and check their contents.

CREATE TABLE Example With a Simple SELECT Statement

This example shows a very simple SQL command for creating a Data Pump format file from the CUSTOMERS table. It selects the entire table and generates a single output file named customers.dmp in the local /expdir directory.

```
CREATE TABLE export_customers
  ORGANIZATION EXTERNAL
  (
    TYPE oracle_datapump
    DEFAULT DIRECTORY expdir
    LOCATION('customers.dmp')
  )
AS SELECT * FROM customers;
```

CREATE TABLE Example With a More Complex SQL SELECT Statement

The next example shows more complexity in the syntax. It joins the CUSTOMERS and COUNTRIES tables on the COUNTRY_ID columns to provide the country names. It also limits the rows to customers in the Americas. The command generates two output files in parallel, named americas1.dmp and americas2.dmp, in the local /expdir directory.

```
CREATE TABLE export_americas
  ORGANIZATION EXTERNAL
  (
    TYPE oracle_datapump
```

```

    DEFAULT DIRECTORY expdir
    LOCATION('americas1.dmp', 'americas2.dmp')
  )
  PARALLEL 2
AS SELECT a.cust_first_name first_name,
        a.cust_last_name last_name,
        a.cust_gender gender,
        a.cust_year_of_birth birth,
        a.cust_email email,
        a.cust_postal_code postal_code,
        b.country_name country
FROM customers a,
     countries b
WHERE a.country_id=b.country_id AND
      b.country_region='Americas'
ORDER BY a.country_id, a.cust_postal_code;

```

Verifying the Contents of the Data Files

You can check the content of the output data files before copying them to Hadoop. The previous `CREATE TABLE` statement created an external table named `EXPORT_AMERICAS`, which you can describe and query the same as any other table.

The `DESCRIBE` statement shows the selection of columns and the modified names:

```

SQL> DESCRIBE export_americas;
Name                               Null?    Type
-----
FIRST_NAME                         NOT NULL VARCHAR2(20)
LAST_NAME                         NOT NULL VARCHAR2(40)
GENDER                            NOT NULL CHAR(1)
BIRTH                             NOT NULL NUMBER(4)
EMAIL                             VARCHAR2(50)
POSTAL_CODE                       NOT NULL VARCHAR2(10)
COUNTRY                           NOT NULL VARCHAR2(40)

```

A `SELECT` statement like the following shows a sample of the data:

```

SELECT first_name, last_name, gender, birth, country
FROM export_americas
WHERE birth > 1985
ORDER BY last_name
FETCH FIRST 5 ROWS ONLY;

```

FIRST_NAME	LAST_NAME	GENDER	BIRTH	COUNTRY
Opal	Aaron	M	1990	United States of America
KaKit	Abeles	M	1986	United States of America
Mitchel	Alambarati	M	1987	Canada
Jade	Anderson	M	1986	United States of America
Roderica	Austin	M	1986	United States of America

Copying the Files into Hadoop

The following commands list the files in the local `expdir` directory, create a Hadoop subdirectory named `customers`, and copy the files to it. The user is connected to Oracle Big Data Appliance as the `oracle` file system user.

```

$ cd /expdir
$ ls americas*.dmp

```

```

americas1.dmp  americas2.dmp
$ hadoop fs -mkdir customers
$ hadoop fs -put *.dmp customers
$ hadoop fs -ls customers
Found 2 items
-rw-r--r--  1 oracle oracle      798720 2014-10-13 17:04 customers/americas1.dmp
-rw-r--r--  1 oracle oracle      954368 2014-10-13 17:04 customers/americas2.dmp

```

Creating a Hive External Table

This HiveQL statement creates an external table using the Copy to BDA SerDes. The **LOCATION** clause identifies the full path to the Hadoop directory containing the Data Pump files:

```

CREATE EXTERNAL TABLE customers
  ROW FORMAT SERDE 'oracle.hadoop.hive.datapump.DPSerDe'
  STORED AS
    INPUTFORMAT 'oracle.hadoop.hive.datapump.DPInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat'
  LOCATION '/user/oracle/customers';

```

The **DESCRIBE** command shows the columns of the **CUSTOMERS** external table.

```

hive> DESCRIBE customers;
OK
first_name      varchar(20)      from deserializer
last_name       varchar(40)      from deserializer
gender          char(1)          from deserializer
birth           int              from deserializer
email           varchar(50)      from deserializer
postal_code     varchar(10)      from deserializer
country         varchar(40)      from deserializer

```

Querying the Data in Hive

The following HiveQL **SELECT** statement shows the same data as the SQL **SELECT** statement from Oracle Database shown in ["Verifying the Contents of the Data Files"](#) on page 8-7. The two queries access copies of the same Data Pump files.

```

SELECT first_name, last_name, gender, birth, country
  FROM customers
 WHERE birth > 1985
 ORDER BY last_name LIMIT 5;

```

```

Total MapReduce jobs = 1
Launching Job 1 out of 1

```

```

.
.
.
OK
Opal      Aaron      M      1990      United States of America
KaKit     Abeles     M      1986     United States of America
Mitchel   Alambarati M      1987     Canada
Jade      Anderson   M      1986     United States of America
Roderica  Austin     M      1986     United States of America

```

Glossary

Apache Flume

A distributed service for collecting and aggregating data from almost any source into a data store such as HDFS or HBase.

See also [Apache HBase](#); [HDFS](#).

Apache HBase

An open-source, column-oriented database that provides random, read/write access to large amounts of sparse data stored in a CDH cluster. It provides fast lookup of values by key and can perform thousands of insert, update, and delete operations per second.

Apache Hive

An open-source data warehouse in CDH that supports data summarization, ad hoc querying, and data analysis of data stored in HDFS. It uses a SQL-like language called HiveQL. An interpreter generates MapReduce code from the HiveQL queries.

By using Hive, you can avoid writing MapReduce programs in Java.

See also [Hive Thrift](#); [HiveQL](#); [MapReduce](#).

Apache Sentry

Integrates with the Hive and Impala SQL-query engines to provide fine-grained authorization to data and metadata stored in Hadoop.

Apache Solr

Provides an enterprise search platform that includes full-text search, faceted search, geospatial search, and hit highlighting.

Apache Spark

A fast engine for processing large-scale data. It supports Java, Scala, and Python applications. Because it provides primitives for in-memory cluster computing, it is particularly suited to machine-learning algorithms. It promises performance up to 100 times faster than MapReduce.

Apache Sqoop

A command-line tool that imports and exports data between HDFS or Hive and structured databases. The name Sqoop comes from "SQL to Hadoop." Oracle R Advanced Analytics for Hadoop uses the Sqoop executable to move data between HDFS and Oracle Database.

Apache YARN

An updated version of MapReduce, also called MapReduce 2. The acronym stands for Yet Another Resource Negotiator.

ASR

Oracle Auto Service Request, a software tool that monitors the health of the hardware and automatically generates a service request if it detects a problem.

See also [OASM](#).

Balancer

A service that ensures that all nodes in the cluster store about the same amount of data, within a set range. Data is balanced over the nodes in the cluster, not over the disks in a node.

CDH

Cloudera's Distribution including Apache Hadoop, the version of Apache Hadoop and related components installed on Oracle Big Data Appliance.

Cloudera Hue

Hadoop User Experience, a web user interface in CDH that includes several applications, including a file browser for HDFS, a job browser, an account management tool, a MapReduce job designer, and Hive wizards. Cloudera Manager runs on Hue.

See also [HDFS](#); [Apache Hive](#).

Cloudera Impala

A massively parallel processing query engine that delivers better performance for SQL queries against data in HDFS and HBase, without moving or transforming the data.

Cloudera Manager

Cloudera Manager enables you to monitor, diagnose, and manage CDH services in a cluster.

The Cloudera Manager agents on Oracle Big Data Appliance also provide information to Oracle Enterprise Manager, which you can use to monitor both software and hardware.

Cloudera Navigator

Verifies access privileges and audits access to data stored in Hadoop, including Hive metadata and HDFS data accessed through HDFS, Hive, or HBase.

Cloudera Search

Provides search and navigation tools for data stored in Hadoop. Based on Apache Solr.

Cloudera's Distribution including Apache Hadoop (CDH)

See [CDH](#).

cluster

A group of servers on a network that are configured to work together. A server is either a master node or a worker node.

All servers in an Oracle Big Data Appliance rack form a cluster. Servers 1, 2, and 3 are master nodes. Servers 4 to 18 are worker nodes.

See [Hadoop](#).

DataNode

A server in a CDH cluster that stores data in HDFS. A DataNode performs file system operations assigned by the NameNode.

See also [HDFS](#); [NameNode](#).

Flume

See [Apache Flume](#).

Hadoop

A batch processing infrastructure that stores files and distributes work across a group of servers. Oracle Big Data Appliance uses Cloudera's Distribution including Apache Hadoop (CDH).

Hadoop Distributed File System (HDFS)

See [HDFS](#).

Hadoop User Experience (Hue)

See [Cloudera Hue](#).

HBase

See [Apache HBase](#).

HDFS

Hadoop Distributed File System, an open-source file system designed to store extremely large data files (megabytes to petabytes) with streaming data access patterns. HDFS splits these files into data blocks and distributes the blocks across a CDH cluster.

When a data set is larger than the storage capacity of a single computer, then it must be partitioned across several computers. A distributed file system can manage the storage of a data set across a network of computers.

See also [cluster](#).

Hive

See [Apache Hive](#).

Hive Thrift

A remote procedure call (RPC) interface for remote access to CDH for Hive queries.

See also [CDH](#); [Apache Hive](#).

HiveQL

A SQL-like query language used by Hive.

See also [Apache Hive](#).

HotSpot

A Java Virtual Machine (JVM) that is maintained and distributed by Oracle. It automatically optimizes code that executes frequently, leading to high performance. HotSpot is the standard JVM for the other components of the Oracle Big Data Appliance stack.

Hue

See [Cloudera Hue](#).

Impala

See [Cloudera Impala](#).

Java HotSpot Virtual Machine

See [HotSpot](#).

JobTracker

A service that assigns tasks to specific nodes in the CDH cluster, preferably those nodes storing the data. MRv1 only.

See also [Hadoop](#); [MapReduce](#).

Kerberos

A network authentication protocol that helps prevent malicious impersonation. It was developed at the Massachusetts Institute of Technology (MIT).

Mahout

Apache Mahout is a machine learning library that includes core algorithms for clustering, classification, and batch-based collaborative filtering.

MapReduce

A parallel programming model for processing data on a distributed system. Two versions of MapReduce are available, MapReduce 1 and YARN (MapReduce 2). The default version on Oracle Big Data Appliance 3.0 and later is YARN.

A MapReduce program contains these functions:

- Mappers: Process the records of the data set.
- Reducers: Merge the output from several mappers.
- Combiners: Optimizes the result sets from the mappers before sending them to the reducers (optional and not supported by all applications).

See also [Apache YARN](#).

MySQL Database

A SQL-based relational database management system. Cloudera Manager, Oracle Data Integrator, Hive, and Oozie use MySQL Database as a metadata repository on Oracle Big Data Appliance.

NameNode

A service that maintains a directory of all files in HDFS and tracks where data is stored in the CDH cluster.

See also [HDFS](#).

Navigator

See [Cloudera Navigator](#).

node

A server in a CDH cluster.

See also [cluster](#).

NodeManager

A service that runs on each node and executes the tasks assigned to it by the ResourceManager. YARN only.

See also [ResourceManager](#); [YARN](#).

NoSQL Database

See [Oracle NoSQL Database](#).

OASM

Oracle Automated Service Manager, a service for monitoring the health of Oracle Sun hardware systems. Formerly named Sun Automatic Service Manager (SASM).

Oozie

An open-source workflow and coordination service for managing data processing jobs in CDH.

Oracle Database Instant Client

A small-footprint client that enables Oracle applications to run without a standard Oracle Database client.

Oracle Linux

An open-source operating system. Oracle Linux 5.6 is the same version used by Exalogic 1.1. It features the Oracle Unbreakable Enterprise Kernel.

Oracle NoSQL Database

A distributed key-value database that supports fast querying of the data, typically by key lookup.

Oracle R Distribution

An Oracle-supported distribution of the R open-source language and environment for statistical analysis and graphing.

Oracle R Enterprise

A component of the Oracle Advanced Analytics Option. It enables R users to run R commands and scripts for statistical and graphical analyses on data stored in an Oracle database.

Pig

An open-source platform for analyzing large data sets that consists of the following:

- Pig Latin scripting language
- Pig interpreter that converts Pig Latin scripts into MapReduce jobs

Pig runs as a client application.

See also [MapReduce](#).

Puppet

A configuration management tool for deploying and configuring software components across a cluster. The Oracle Big Data Appliance initial software installation uses Puppet.

The Puppet tool consists of these components: puppet agents, typically just called puppets; the puppet master server; a console; and a cloud provisioner.

See also [puppet agent](#); [puppet master](#).

puppet agent

A service that primarily pulls configurations from the puppet master and applies them. Puppet agents run on every server in Oracle Big Data Appliance.

See also [Puppet](#); [puppet master](#)

puppet master

A service that primarily serves configurations to the puppet agents.

See also [Puppet](#); [puppet agent](#).

ResourceManager

A service that assigns tasks to specific nodes in the CDH cluster, preferably those nodes storing the data. YARN only.

See also [Hadoop](#); [YARN](#).

Search

See [Cloudera Search](#).

Sentry

See [Apache Sentry](#).

Solr

See [Apache Solr](#).

Spark

See [Apache Spark](#).

Sqoop

See [Apache Sqoop](#).

table

In Hive, all files in a directory stored in HDFS.

See also [HDFS](#).

TaskTracker

A service that runs on each node and executes the tasks assigned to it by the JobTracker service. MRv1 only.

See also [JobTracker](#).

Whirr

Apache Whirr is a set of libraries for running cloud services.

YARN

See [Apache YARN](#).

ZooKeeper

A MapReduce 1 centralized coordination service for CDH distributed processes that maintains configuration information and naming, and provides distributed synchronization and group services.

A

- access drivers, 6-2, 8-3
- ACCESS PARAMETERS Clause
 - syntax, 7-5
- ACCESS PARAMETERS clause, 6-18
 - special characters, 7-6
 - syntax rules, 7-5
- activity reports, 2-30
- ALL_HIVE_COLUMNS view, 6-6, 7-26
- ALL_HIVE_DATABASES view, 7-24
- ALL_HIVE_TABLES view, 6-5, 7-25
- Apache Sentry, 2-27
- application adapters, 1-8
- applications
 - data pull, 4-1
 - data push, 4-2
- array overflows, 7-19
- Audit Vault
 - plug-in configuration, 6-3
- Audit Vault plug-in, 2-29
- auditing data collected from services, 2-29
- authentication, 3-1
- authorization, 2-27
- autoAnalyze configuration property, 5-11, 5-20
- autoAnalyze property, 5-7
- autoBalance configuration property, 5-11, 5-20
- Automated Service Manager
 - See* OASM

B

- BALANCER_HOME environment variable, 5-3
- bddiag utility, 2-31
- Berkeley DB, 1-5
- best practices, 5-1
- big data description, 1-1
- binary overflows, 7-19
- business intelligence, 1-3, 1-5, 1-9
- byteWeight configuration property, 5-21

C

- catalog views, 7-23
- CDH
 - about, 1-3

- diagnostics, 2-31
- file system, 1-5
- remote client access, 3-2
- security, 3-1
- version, 2-8
- character overflows, 7-19
- chopped keys, 5-20
- chunking files, 1-5
- client access
 - HDFS cluster, 3-3
 - HDFS secured cluster, 3-4
 - Hive, 3-6
- client configuration, 3-2
- Cloudera Manager
 - about, 2-3
 - accessing administrative tools, 2-4
 - connecting to, 2-3
 - effect of hardware failure on, 2-17
 - software dependencies, 2-17
 - starting, 2-3
 - UI overview, 2-3
 - version, 2-8
- Cloudera's Distribution including Apache Hadoop
 - See* CDH
- clusters, definition, 1-3
- column mapping, 7-10
- common directory, 6-20
- com.oracle.bigdata.colmap, 7-10
- com.oracle.bigdata.datamode, 7-11
- com.oracle.bigdata.erroropt, 7-12
- com.oracle.bigdata.fields, 7-14
- com.oracle.bigdata.fileformat, 7-16
- com.oracle.bigdata.log.exec, 7-17
- com.oracle.bigdata.log.qc, 7-18
- com.oracle.bigdata.overflow, 7-19
- com.oracle.bigdata.rowformat, 7-20
- com.oracle.bigdata.tablename, 7-22
- confidence configuration property, 5-20
- Copy to BDA utility, 8-1
- Counting Reducer, 5-2
- CREATE TABLE ORGANIZATION EXTERNAL
 - syntax, 6-16, 8-3
- CREATE TABLE statement
 - generating automatically for Hive, 7-3
- CREATE_EXTDDL_FOR_HIVE function, 6-6
 - syntax, 7-3

D

- data dictionary views, 7-23
- data mode, 7-11
- data replication, 1-5
- data skew, 5-1
- data source name, 7-22
- data type conversion (Big Data SQL), 6-18
- data types (HDFS), 7-14
- DataNode, 2-15
- dba group, 2-26
- DBA_HIVE_COLUMNS view, 7-29
- DBA_HIVE_DATABASES view, 7-27
- DBA_HIVE_TABLES view, 7-28
- DBMS_HADOOP package, 6-6, 7-2
- DBMS_OUTPUT package, 6-6
- DEFAULT DIRECTORY clause, 6-17
- delimited text files, 7-20
- diagnostics, collecting, 2-31
- disks, 2-15
- dnsmasq service, 4-4
- duplicating data, 1-5

E

- emcli utility, 2-2
- enableSorting configuration property, 5-20
- encryption, 2-27
- engineered systems, 1-3
- error handling, 7-12
- error handling (Big Data SQL), 6-19
- Exadata Database Machine, 1-3
- Exadata InfiniBand connections, 4-2
- Exalytics In-Memory Machine, 1-3
- External table clause, 6-16
- external tables, 1-8
 - about, 6-2, 8-3

F

- failover
 - JobTracker, 2-13
 - NameNode, 2-12
- feedbackDir configuration property, 5-22
- field extraction, 7-20
- field names, 7-14
- files, recovering HDFS, 3-10
- first NameNode, 2-16
- Flume, 2-9, 2-26
- ftp.oracle.com, 2-31

G

- groups, 2-26, 3-8

H

- Hadoop Distributed File System
 - See* HDFS
- hadoop group, 3-8
- Hadoop log files, 7-17

- Hadoop version, 1-3
- HADOOP_CLASSPATH environment variable, 5-3, 5-17
- HADOOP_USER_CLASSPATH_FIRST environment variable, 5-3
- HBase, 2-9, 2-26
- HDFS
 - about, 1-4, 1-5
 - auditing, 2-29
 - user identity, 2-26
- HDFS files, 6-14
- help from Oracle Support, 2-31
- Hive, 2-26
 - about, 1-5
 - auditing, 2-29
 - client access, 3-6
 - node location, 2-17
 - software dependencies, 2-17
 - tables, 3-8
 - user identity, 2-26
- Hive columns, 7-26
- Hive data
 - access from Oracle Database, 6-5
- Hive databases, 7-24
- hive group, 3-8
- Hive table sources, 7-22
- Hive tables, 7-25
- Hive views, 7-23
- HiveQL, 1-5
- HotSpot
 - See* Java HotSpot Virtual Machine
- Hue, 2-17
 - user identity, 2-26
 - users, 3-8

I

- Impala, 2-9
- InfiniBand connections to Exadata, 4-2
- InfiniBand network configuration, 4-1
- inputFormat.mapred.* configuration
 - properties, 5-20
- installing CDH client, 3-2

J

- Java HotSpot Virtual Machine, 2-8
- Job Analyzer, 5-2, 5-4
- job duration, 5-1
- jobconfPath property, 5-19
- jobHistoryPath configuration property, 5-19
- JobTracker
 - failover, 2-13
 - security, 3-1
 - user identity, 2-26
- JobTracker node, 2-17

K

- Kerberos authentication, 3-1
- Kerberos commands, 3-1

- Kerberos user setup, 3-9
- key chopping, 5-2
- keyLoad.minChopBytes configuration property, 5-21
- keys, assigning to reducers, 5-2
- key-value database, 1-5
- keyWeight configuration property, 5-22
- knowledge modules, 1-8

L

- linearKeyLoad properties, 5-8
- linearKeyLoad.* configuration properties, 5-21
- Linux
 - disk location, 2-15
 - installation, 2-8
- load, 5-1
- Load Balancer, 5-2
- loading data, 1-8
- LOCATION clause, 6-17
- log files, 7-18
- login privileges, 3-9

M

- mapper workload, 5-2
- mapred configuration properties, 5-18
- mapred user, 2-26
- mapred.map.tasks configuration property, 5-21
- MapReduce, 1-4, 1-7, 2-29, 3-1, 3-8
- mapreduce configuration properties, 5-18
- map.tasks property, 5-20
- maxLoadFactor configuration property, 5-22
- maxSamplesPct configuration property, 5-23
- max.split.size configuration property, 5-21
- minChopBytes configuration property, 5-21
- minSplits configuration property, 5-23
- monitoring activity, 2-30
- multitrack clusters
 - service locations, 2-11
- MySQL Database
 - about, 2-17
 - port number, 2-28
 - user identity, 2-26
 - version, 2-8

N

- NameNode, 3-1
 - first, 2-16
- NameNode failover, 2-12
- Navigator, 2-9
- NoSQL databases
 - See* Oracle NoSQL Database
- numThreads configuration property, 5-23

O

- OASM, port number, 2-28
- ODI
 - See* Oracle Data Integrator

- oinstall group, 2-26, 3-8
- on-disk encryption, 2-27
- Oozie, 2-17
 - auditing, 2-29
 - software dependencies, 2-17
 - software services, 2-26
 - user identity, 2-26
- openib.conf file, 4-4
- operating system users, 2-26
- Oracle Audit Vault and Database Firewall, 2-28
 - plug-in configuration, 6-3
- Oracle Automated Service Manager
 - See* OASM
- Oracle Big Data SQL
 - access drivers, 6-2
 - data type conversion, 6-18
 - general description, 1-7, 6-1
 - installation changes on Oracle Exadata Machine, 6-20
 - security, 6-3
- Oracle Data Integrator
 - about, 1-8
 - node location, 2-17
 - software dependencies, 2-17
 - version, 2-8
- Oracle Data Integrator agent, 2-28
- Oracle Database
 - access to Hive data, 6-5
 - HDFS file access, 6-14
- Oracle Database Instant Client, 2-8
- Oracle Exadata Database Machine, 1-3, 4-1
- Oracle Exadata Machine
 - Big Data SQL installation changes, 6-20
- Oracle Exalytics In-Memory Machine, 1-3
- Oracle Linux
 - about, 1-3
 - relationship to HDFS, 1-4
 - version, 2-8
- Oracle Loader for Hadoop, 1-8, 2-8
- Oracle NoSQL Database
 - about, 1-5, 1-8
 - port numbers, 2-28
 - version, 2-8
- Oracle R Advanced Analytics for Hadoop, 1-8, 2-8
- Oracle R Enterprise, 1-9
- Oracle SQL Connector for HDFS, 1-8
- Oracle Support, creating a service request, 2-31
- oracle user, 2-26, 3-8
- Oracle XQuery for Hadoop, 1-8, 2-8
- ORACLE_HDFS access driver, 6-14
- ORACLE_HIVE
 - access parameters, 7-9
- ORACLE_HIVE access driver, 6-7
- ORACLE_HIVE examples, 6-7
- oracle.hadoop.balancer.* configuration properties, 5-20
- oracle.hadoop.balancer.autoAnalyze configuration property, 5-11
- oracle.hadoop.balancer.autoAnalyze property, 5-7
- oracle.hadoop.balancer.autoBalance configuration

- property, 5-11
- oracle.hadoop.balancer.Balancer class, 5-15
- oracle.hadoop.balancer.KeyLoadLinear class, 5-22
- oracle.hadoop.balancer.linearKeyLoad.*
 - properties, 5-8
- ORC files, 7-16
- out of heap space errors, 5-14
- overflow handling, 7-19

P

- Parquet files, 7-16
- parsing HDFS files, 7-20
- partitioning, 2-15, 5-2
- Perfect Balance
 - application requirements, 5-2
 - basic steps, 5-3
 - description, 5-1
- planning applications, 1-3
- PL/SQL packages, 7-2
- port map, 2-28
- port numbers, 2-27, 2-28
- pulling data into Exadata, 4-1
- puppet
 - port numbers, 2-28
 - security, 2-28
 - user identity, 2-26
- puppet master
 - node location, 2-16
- pushing data into Exadata, 4-2
- PUT_LINE function, 6-6

R

- R Connector
 - See* Oracle R Advanced Analytics for Hadoop
- R distribution, 2-8
- R language support, 1-9
- range partitioning, 5-2
- RC files, 7-16
- recovering HDFS files, 3-10
- reducer load, 5-1
- REJECT LIMIT clause, 6-18
- remote client access, 3-2, 3-6
- replicating data, 1-5
- report.overwrite configuration property, 5-23
- reportPath configuration property, 5-23
- resource management, 2-10, 2-24
- row format description, 7-16
- row formats, 7-20
- rowWeight configuration property, 5-22
- rpc.statd service, 2-28

S

- SDP listener configuration, 4-5
- SDP over InfiniBand, 4-1
- SDP, enabling on Exadata, 4-4
- Search, 2-9
- security, 2-25
- Sentry, 2-27

- sequence files, 7-16
- SerDe parsing, 7-20
- service requests, creating for CDH, 2-31
- service tags, 2-28
- services
 - auditing, 2-29
 - node locations, 2-10
- skew, 5-1
- SmartScan, 6-2
- SmartScan mode, 7-11
- Sockets Direct Protocol, 4-1
- software components, 2-8
- software framework, 1-3
- software services
 - node locations, 2-10
 - port numbers, 2-28
- source name, 7-22
- Spark, 2-9
- Sqoop, 2-9, 2-26
- ssh service, 2-28
- static data dictionary views, 7-23
- struct overflows, 7-19
- svctag user, 2-26

T

- tables, 1-8, 3-8
- TaskTracker
 - user identity, 2-26
- text files, 7-16
- text overflows, 7-19
- tmpDir configuration property, 5-24
- tools.* configuration properties, 5-19
- trash facility, 3-10
- trash facility, disabling, 3-12
- trash interval, 3-11
- troubleshooting CDH, 2-31
- TYPE clause, 6-17

U

- union overflows, 7-19
- uploading diagnostics, 2-31
- useClusterStats configuration property, 5-24
- useMapreduceApi configuration property, 5-24
- user access from Oracle Database, 6-19
- user accounts, 3-8
- user groups, 3-8
- USER_HIVE_COLUMNS view, 7-32
- USER_HIVE_DATABASES view, 7-30
- USER_HIVE_TABLES view, 7-31
- users
 - Cloudera Manager, 2-4
 - operating system, 2-26

W

- writeKeyBytes configuration property, 5-19

X

xinetd service, 2-28

XQuery connector

See Oracle XQuery for Hadoop

Y

YARN support, 1-7

Z

ZooKeeper, 2-26

